

IBM Content Manager OnDemand



Windows Client Customization Guide

IBM Content Manager OnDemand



Windows Client Customization Guide

Note

Before using this information and the product that it supports, read the information in Appendix F, "Notices" on page 403.

Second Edition (September 2002)

This edition replaces SC27-0837-00.

© **Copyright International Business Machines Corporation 1996, 2002. All rights reserved.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Contents	iii
About this publication	xi
Who should use this publication	xi
How this publication is organized	xi
Where to find more information	xii
Where to get product support	xii
How to send your comments	xii
<hr/>	
The OnDemand OLE Control	1
Overview of OnDemand OLE Control	3
Viewing Multiple Documents for a Single Folder	3
Header File	3
Return Code	4
Methods	5
AboutBox	6
ActivateFolder	7
AnnotateDoc	8
CancelOperation	10
ChangePassword	11
ClearFolderSearchFields	13
CloseAllFolders	14
CloseDoc	15
CloseFolder	16
CopyBitmap	17
CopyDocPagesToFile	18
CopyText	20
DeleteDoc	21
FindStringInDoc	22
GetAnnotationForDoc	26
GetAnnotationStatus	28
GetControlId	30
GetDocAnnotation	32
GetDocBackgroundColor	34
GetDocCurrentPage	36
GetDocDisplayValue	37
GetDocDisplayValues	39
GetDocImageColor	42
GetDocImageIntensity	44
GetDocNumPages	46
GetDocRotation	47
GetDocScrollPositions	49

GetDocType	51
GetDocZoom	53
GetFolderDisplayFieldName	55
GetFolderDisplayFieldNames	58
GetFolderFieldName	60
GetFolderFieldNames	61
GetFolderName	63
GetFolderNames	65
GetFolderSearchFieldName	67
GetFolderSearchFieldNames	70
GetNumDocAnnotations	73
GetNumDocsInList	75
GetNumFolderDisplayFields	78
GetNumFolderFields	81
GetNumFolders	85
GetNumFolderSearchFields	88
GetNumServerPrinters	91
GetNumServers	94
GetServerName	97
GetServerNames	99
GetServerPrinter	101
GetServerPrinterInfo	103
GetTypeForDoc	105
IsDocHorzScrollRequired	107
Logoff	111
Logon	112
OnSysColorChange	115
OpenDoc	116
OpenFolder	120
PrintDoc	123
RetrieveDoc	127
ScrollDocHorz	130
ScrollDocVert	135
SearchFolder	139
SetDefaultFolderSearchFields	144
SetDocBackgroundColor	145
SetDocCurrentPage	146
SetDocImageColor	148
SetDocImageIntensity	149
SetDocRotation	150
SetDocZoom	151
SetFolderCloseMemoryRelease	153
SetFolderSearchFieldData	154
SetLogonReturnOnFailure	159
SetRightButtonMenu	160
SetSelectionMode	163
SetServerPrinterData	165
SetUserMessageMode	167

ShowFolder	168
ShowWaitCursorDuringCancelableOperation	170
StoreDoc	171
UpdateDoc	174
UndoFind	176
WasOperationCancelled	177
OLE Events	179
FolderSearchCompleted	179
FolderClosed	179
DocOpened	179
DocClosed	179
AreaSelected	179
AreaDeselected	180
UserCommand(long CommandID)	180

Windows 32-bit GUI Customization Guide 181

OnDemand Customization Overview 183

Command Line	185
Starting OnDemand 32-bit client	185
Parameter Syntax	185
Parameters	185
Product Title — /T name	185
Example	185
Logon Server Name — /S name	186
Example	186
Logon User ID — /U id	186
Example	186
Logon Password — /P password	186
Example	186
Change Password — /C new password	186
Example	186
Folder Name — /F name	186
Example	186
Maximum Open Folders — /O number	187
Example	187
Window Placement — /W placement	187
Example	187
Enable DDE Interface — /I number,path,resid	187
Example	187
Disable Exit — /K	187
Example	187
Disable Logoff or Password Change — /X	188
Example	188
Disable Update Servers — /Y	188
Example	188

Disable Close Folder — /Z	188
Example	188
Disable Anticipation — /V	188
Example	188
Disable User Confirmation — /B	188
Example	188
Free Memory When Folder Closed — /Q	188
Example	188
Language Path — /1	188
Example	189
Dynamic Data Exchange (DDE) and DDE Management Library	191
Invoking OnDemand 32-bit from Another Windows Application	191
OnDemand Invocation and DDEML Initialization	192
DDEML Termination	194
DDEML Transactions	195
OnDemand DDE Commands	198
ACTIVATE_DOC	199
ACTIVATE_FOLDER	200
ANNOTATE_DOC	201
ARRANGE_DOCS	203
CHANGE_PASSWORD	204
CLEAR_FIELDS	205
CLOSE_ALL_DOCS	206
CLOSE_ALL_FOLDERS	207
CLOSE_DOC	208
CLOSE_FOLDER	209
COPY_DOC_PAGES	210
DELETE_DOC	212
DESELECT_DOC	213
DISABLE_SWITCH	214
ENABLE_SWITCH	215
EXIT	216
GET_DISPLAY_FIELDS	217
GET_DOC_VALUES	218
GET_FOLDER_FIELDS	220
GET_FOLDERS	221
GET_NUM_DOCS_IN_LIST	222
GET_NUM_DOC_PAGES	223
GET_PRINTERS	224
GET_QUERY_FIELDS	225
GET_SERVERS	226
LOGOFF	227
LOGON	228
OPEN_DOC	230
OPEN_FOLDER	232
PRINT_DOC	237
RESTORE_DEFAULTS	240

RETRIEVE_DOC	241
SEARCH_FOLDER	243
SELECT_DOC	244
SET_FIELD_DATA	245
SET_FOCUS	247
SET_HELP_PATH	248
SET_USER_MSG_MODE	249
SHOW_WINDOW	250
STORE_DOC	251
UPDATE_DOC	254
Return Codes	256
DDEML Advise Loop	258
External Applications and Dynamic Link Libraries	261
Related Documents	269
Product Information File	273
Document Audit Facility	275
Overview	275
Creating the DAF control file	275
The AUDIT section	276
The folder section	276
Defining the report	277
Defining the application group	277
Defining the application	278
Defining the Folder	278
Controlling access to the DAF	278
Using the DAF	279
Integration with Monarch Version 5	281
Before you begin	282
Configuring the client	282
Adding the Registry key	283
Exporting the Registry key	287
Using multiple Monarch model files	287
Configuring Setup	289
Copying client software	289
Adding subdirectories	289
Copying Monarch files	290
Sharing the installation directory	290
Running Setup	290
Running Monarch from OnDemand	291
Upgrading your OnDemand client	291
Installing client software on a network	293
Sharing OnDemand clients among multiple users	293

Installation directories	294
Distribution install	294
Overview	295
Distributing Adobe software	295
Copying OnDemand software to the server	295
Distributing user-defined files	295
Multiple user install	296
Overview	296
Installing Adobe software	296
Installing the OnDemand client on the server	296
Sharing user-defined files	297
Node install	297
Overview	297
Installing Adobe software	297
Installing the client	298
Distributing user-defined files	299
Overview	299
Copying OnDemand client software to the server	300
Adding subdirectories	300
Storing user-defined files on the server	301
Installing the OnDemand client	301
Using response files	303
Introduction	303
Format of a response file	303
Creating a response file	303
Installing software using a response file	304
Verifying software installation	304
Using a response file to install OnDemand software	304
Mapping AFP fonts	307
When you need to map fonts	307
Files supplied for mapping fonts	308
Steps for mapping fonts	309
Syntax rules for font definition files	310
Coded Font file	310
Coded Font file rules	311
Character Set Definition file	311
Character Set Definition file rules	314
Code Page Definition file	314
Code Page Definition file rules	315
Code Page Map files	315
Code Page Map file rules	316
Code Page Map file REXX program for building a Code Page Map file	317
Setting up to build a Code Page Map file	317
Alias file	318
Alias File Rules	319

Support for TrueType fonts	320
TrueType Fonts	321
TrueType Font Substitution Problems	321
Appendix A. Microsoft Visual Basic 5.0 DDE Program Sample	323
Global Variables Used by the Demo	324
Entry Point for the Demo	325
Appendix B. Microsoft VC++ 5.0 DDE Program Sample	345
Appendix C. Microsoft Visual Basic 5.0 OLE Program Sample	363
Global Variables Used by the Demo	364
Appendix D. Microsoft VC++ 5.0 OLE Program Sample	381
Appendix E. Accessibility features	401
Keyboard input and navigation	401
Keyboard input	401
Keyboard focus	401
Features for accessible display	401
High-contrast mode	401
Font settings	401
Non-dependence on color	402
Alternative alert cues	402
Compatibility with assistive technologies	402
Accessible documentation	402
Appendix F. Notices	403
Trademarks and Service Marks	405
Index	407

About this publication

This book contains information about OLE (Object Linking and Embedding) Control and how to customize IBM Content Manager OnDemand (OnDemand) by specifying command line parameters, by invoking and manipulating OnDemand from another Windows application with the Dynamic Data Exchange (DDE) interface, or by creating a Product Information File. This book also contains information that administrators can use to distribute the client software and files to multiple users over a network.

Note: The term *Windows* refers to Windows 2000, Windows 98, Windows NT 4.0 with SP5 or later, and Windows XP. The term *server* refers to systems that are running the OnDemand for iSeries Version 5 Release 2, OnDemand for Multiplatforms Version 7.1, and OnDemand for z/OS and OS/390, Version 7.1 software.

Who should use this publication

This book is of primary interest to programmers who want to integrate OnDemand with other Windows applications and administrators that are responsible for installing and distributing software products.

How this publication is organized

This book contains the following sections:

- “The OnDemand OLE Control” on page 1 contains an overview of OnDemand OLE Control and describes the methods available for an OnDemand OLE Control.
- “OnDemand Customization Overview” on page 183 contains an overview of how you can customize the OnDemand client.
- “Command Line” on page 185 describes how to start the client, the parameter syntax rules used for the command line parameters, and the parameters recognized by OnDemand Windows Client.
- “Dynamic Data Exchange (DDE) and DDE Management Library” on page 191 describes how to use Dynamic Data Exchange (DDE) with OnDemand
- “External Applications and Dynamic Link Libraries” on page 261 describes menu and toolbar extensions that allow an end user to invoke another Windows application or execute a function in a Dynamic Link Library (DLL).
- “Related Documents” on page 269 describes menu and toolbar extensions that allow an end user to retrieve and view a document related to the document currently being viewed.
- “Product Information File” on page 273 describes how to use the Product Information File (PIF) to customize the OnDemand application title and appearance of the “About” dialog box.
- “Document Audit Facility” on page 275 describes the Document Audit Facility (DAF). You can use the DAF to audit documents with the Windows client.
- “Integration with Monarch Version 5” on page 281 describes how to configure the client to work with the Monarch data mining software.
- “Installing client software on a network” on page 293 provides information about installing the client software to be shared by multiple users over a network.

- “Distributing user-defined files” on page 299 describes how to configure the client installation program to distribute user-defined files.
- “Using response files” on page 303 provides information to help you automate the client installation process.
- “Mapping AFP fonts” on page 307 provides information to help you map the Advanced Function Presentation (AFP) fonts your documents were created with to fonts that can be displayed by the client.
- The Appendixes provide the following sample programs:
 - Appendix A, “Microsoft Visual Basic 5.0 DDE Program Sample” on page 323
 - Appendix B, “Microsoft VC++ 5.0 DDE Program Sample” on page 345
 - Appendix C, “Microsoft Visual Basic 5.0 OLE Program Sample” on page 363
 - Appendix D, “Microsoft VC++ 5.0 OLE Program Sample” on page 381

Where to find more information

OnDemand includes a complete set of information to help you plan for, install, administer, and use the system. OnDemand product documentation is available by following the **Library** link from the product Web site at:

<http://www.ibm.com/software/ondemand/>

Where to get product support

Product support is available on the Web. Click **Support** from the product Web site at:

<http://www.ibm.com/software/ondemand/>

How to send your comments

Your feedback helps IBM to provide quality information. Please send any comments that you have about this publication or other OnDemand documentation. You can use either of the following methods to provide comments:

- Send your comments from the Web. Visit the IBM Data Management Online Reader's Comment Form (RCF) page at:
<http://www.ibm.com/software/data/rcf>
- Send your comments by e-mail to:
ondemand@us.ibm.com
- Mail your comments by using the reader's comment form provided at the back of this publication.

Be sure to include the name of the product, the version number of the product, and the name of the book. If you are commenting on specific text, please include the location of the text (for example, a chapter and section title, a table number, a page number, or a help topic title).

The OnDemand OLE Control

Overview of OnDemand OLE Control

Note: In order to use this part of the book, you should know how to embed OLE controls in an application.

OnDemand makes an OLE (Object Linking and Embedding) Control available for displaying documents from the OnDemand database. The OLE Control is implemented in ARSOLE.OCX. During OnDemand installation, this file is placed in the same directory as the other OnDemand executables, and the OLE Control is registered with the Windows system. To run your container application from any directory other than where OnDemand was installed, you need to add the OnDemand directory to your path.

The following rules apply to the use of these controls:

- Each control can display only one document at a time. A document must be closed before another can be displayed.
- Scroll bars to control scrolling of the document data are the responsibility of the container application. These must appear outside the OLE Control window. The OLE Control provides methods to direct the scrolling of the document data. Use of these methods is made easier if the scroll bar ranges are set to ARS_OLE_SCROLL_RANGE.
- Multiple folders may be open simultaneously, but only one of these will be the active folder. The OLE Control provides methods to open, close, and activate a folder.
- The container application can completely control logon, open folder, search folder, close folder, and open document operations or it can cause the normal OnDemand dialog boxes to be used for these operations.

Viewing Multiple Documents for a Single Folder

Each OnDemand OLE Control has a unique run-time control id. This control id can be retrieved with the GetControlId method.

Control ids allow multiple OnDemand OLE Controls to simultaneously display documents from a single folder document list. This avoids the overhead of multiple logon, open folder, and search folder operations.

A given application can include more than one OnDemand OLE Control. That application could use one of those controls to logon, open a folder, and search the folder to create a list of documents. If the control id for that control is made available, the other controls could reference it when using the OpenDoc method and display documents from the single document list.

Header File

The ARSOLEEX.H header file contains definitions of symbolic values used in the OLE control methods described below. It can be included in C/C++ implementations or used as a reference for other languages.

The header file is installed into the INC subdirectory of the OnDemand installation directory. That directory can be added to the include file path or the file can be copied to another directory.

Return Code

Most OnDemand OLE Control methods return a short value. A list of the return code values, such as ARS_OLE_RC_SUCCESS, can be found in ARSOLEEX.H.

Methods

The following sections describe the methods available for an OnDemand OLE Control.

AboutBox

void **AboutBox**()

Description: The OnDemand About Box is displayed.

Return Value: None.

Example. The following example displays the OnDemand About Box.

C/C++ Example

```
CArs01e * pArsCtrl;  
  
.  
.  
  
pArsCtrl->AboutBox( );  
  
.  
.
```

Visual Basic Example

```
.  
.  
.  
  
Ars01e.AboutBox  
  
.  
.  
.
```

ActivateFolder

short **ActivateFolder**(
 char * **pFolderName**)

Parameters: **pFolderName**

Points to a null-terminated character string containing the name of the folder to be activated.

Description: The named folder becomes the active folder.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: OpenFolder, CloseFolder, CloseAllFolders

Example The following example activates a folder.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
  
.  
.  
  
rc = pArsCtrl->ActivateFolder( "Henry's Folder" );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
.  
  
rc = ArsOle.ActivateFolder("Henry's Folder")  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.  
.
```

AnnotateDoc

```
short AnnotateDoc(  
    long      Index,  
    char *    pText,  
    long      Page,  
    boolean   Public,  
    boolean   CanBeCopied )
```

Parameters:

Index

Specifies the zero-based index of a document within the document list of the active folder. If this value is less than zero, the annotation is associated with the open document.

pText

Points to a null-terminated character string containing the text of the annotation. If the text contains more than 32,700 characters, it is truncated.

Page

Specifies the page number to be associated with the annotation.

Public

Indicates whether the annotation is public.

CanBeCopied

Indicates whether the annotation may be copied to other servers.

Description: An annotation is created in the database and associated with the specified document.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: OpenDoc

Example

The following example creates an annotation for a document.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
  
.  
.  
  
rc = pArsCtrl->AnnotateDoc( 3, "This is the text.", 5, TRUE, FALSE );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
.  
  
rc = ArsOle.AnnotateDoc(3, "This is the text.", 5, True, False)  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.  
.
```

CancelOperation

short **CancelOperation**()

Description: Cancels an operation that was started by a SearchFolder, OpenDoc, or RetrieveDoc method.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: SearchFolder, OpenDoc, WasOperationCancelled, ShowWaitCursorDuringCancelableOperation

Example: The following example cancels an operation.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
.  
.  
rc = pArsCtrl->CancelOperation( );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
  
rc = ArsOle.CancelOperation ()  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

ChangePassword

```
short ChangePassword(  
    char *    pCurrentPassword,  
    char *    pNewPassword1,  
    char *    pNewPassword2)
```

Parameters: **pCurrentPassword**

Specifies the users current password.

pNewPassword1

Specifies the users new password.

pNewPassword2

Specifies the users new password again. This is for verification.

Description: OnDemand changes the logon password for the current user.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: Logon

Example:

C/C++ Example

```

CArsOle * pArsCtrl;
short rc;

.
.

rc = pArsCtrl->ChangePassword( "tt1sd",
                                "sfd45r",
                                "sfd45r" );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

```

```

.
.

```

Visual Basic Example

```

Dim rc As Integer

```

```

.
.

rc = ArsOle.ChangePassword ( "tt1sd", _
                             "sfd45r", _
                             "sfd45r" )
if rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

```

.
.

```

ClearFolderSearchFields

short **ClearFolderSearchFields**()

Description: The search fields for the active folder are cleared.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: OpenFolder, SearchFolder ,*

Example: The following example clears the search fields for the active folder.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
  
.  
.  
  
rc = pArsCtrl->ClearFolderSearchFields( );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
.  
  
rc = ArsOle.ClearFolderSearchFields()  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.  
.
```

CloseAllFolders

short **CloseAllFolders**()

Description: All open folders are closed. This causes all open documents associated with the folders to be closed.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: OpenFolder, CloseFolder

Example. The following example closes all folders.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
  
.  
.  
  
rc = pArsCtrl->CloseAllFolders( );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
.  
  
rc = ArsOle.CloseAllFolders()  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.  
.
```

CloseDoc

short **CloseDoc**()

Description: The open document is closed and the control window is repainted with a white background.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

Seealso: OpenDoc

Example: The following example closes a document.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
  
.  
.  
  
rc = pArsCtrl->CloseDoc( );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
.  
  
rc = ArsOle.CloseDoc()  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.  
.
```

CloseFolder

short **CloseFolder**()

Description: The active folder is closed. This causes all open documents associated with the folder to be closed. If any other folders are open, one of them becomes the active folder. If more than one other folder is open, the container application should invoke the **ActivateFolder** method to specify the folder which is to be active.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: **OpenFolder**, **CloseAllFolders**

Example: The following example closes the active folder.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
  
.  
.  
  
rc = pArsCtrl->CloseFolder( );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
  
rc = ArsOle.CloseFolder()  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

CopyBitmap

short **CopyBitmap**()

Description: Copies a selected area of the document to the clipboard in bitmap format.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: CopyText, SetSelectionMode

Example: The following example copies a selected area of the document to the clipboard in bitmap format.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
.  
.  
rc = pArsCtrl->CopyBitmap( );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
  
rc = ArsOle.CopyBitmap ()  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

CopyDocPagesToFile

```
short CopyDocPagesToFile(  
    char *    pPath,  
    long      Page,  
    boolean   AsIs )
```

Parameters: **Page**

Specifies the page number to be copied. If this parameter is less than or equal to zero, the entire document is copied.

pPath

Points to a null-terminated character string containing the fully-qualified path of a file to which the data is to be copied. If the file already exists, the data is appended to the file.

AsIs (for AFP and line data only)

If non-zero, indicates that the data is to be copied “asis”; if zero, that it is to be converted to ASCII.

Description: The data for the page or pages of the open document is copied to the specified file.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: SearchFolder, GetNumDocsInList

Example: The following example copies page 5 of the open document to a file in ASCII format.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
  
.  
.  
  
rc = pArsCtrl->CopyDocPagesToFile( "C:\\FILES\\MYDATA.FIL", 5, FALSE );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
.  
.
```

Visual Basic Example


```

Dim rc As Integer

.
.

rc = ArsOle.CopyDocPagesToFile("C:\FILES\MYDATA.FIL", 5, False)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.

```

CopyText

short **CopyText**()

Description: Copies a selected area of the document to the clipboard in text format.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: CopyBitmap, SetSelectionMode

Example: The following example copies a selected area of the document to the clipboard in text format.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
.  
.  
rc = pArsCtrl->CopyText( );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
  
rc = ArsOle.CopyText ()  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

DeleteDoc

short **DeleteDoc**(
long **DocIndex**)

Parameters: **DocIndex**

Specifies the zero-based relative document number within the document list of the active folder.

Description: OnDemand deletes the specified document from the database. Since the document numbers may have changed, information from a previous GetDocDisplayValues method may no longer be valid.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetNumDocsInList

Example: The following example deletes the first document in the document list of the active folder.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
  
.  
.  
  
rc = pArsCtrl->DeleteDoc( 0 );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
  
rc = ArsOle.DeleteDoc (0)  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

FindStringInDoc

```
short FindStringInDoc(  
    char *      pString,  
    long        Page,  
    short       Type,  
    boolean     CaseSensitive,  
    VARIANT *   pFound,  
    VARIANT *   pHorzPosition,  
    VARIANT *   pVertPosition )
```

Parameters: pString

Points to a null-terminated character string containing the text to be found.

Page

Specifies the page on which the search is to begin. If **Type** specifies ARS_OLE_FIND_PREV or ARS_OLE_FIND_NEXT, the page must be the same as that on which a current find is highlighted.

Type

Specifies the type of find operation. This must be one of the following type values found in ARSOLEEX.H:

```
ARS_OLE_FIND_FIRST  
ARS_OLE_FIND_PREV  
ARS_OLE_FIND_NEXT
```

CaseSensitive

If non-zero, indicates that the search should be case sensitive; if zero, that the case should be ignored.

pFound

Points to a variable to receive a found/not found indication. On return, this variable is set to type VT_I2.

pHorzPosition

Points to a variable to receive the new horizontal scroll position. On return, this variable is set to type VT_I2.

pVertPosition

Points to a variable to receive the new vertical scroll position. On return, this variable is set to type VT_I2.

Description: A search is conducted for the text string beginning on the specified page. The variable pointed to by **pFound** is set to non-zero if the search succeeds; zero if it fails. If the search is successful, the page on which the string is found is made the current page, the string is highlighted and scrolled into view, and the new scroll positions are returned in the specified variables. The scroll positions assume that the scroll ranges have been set to ARS_OLE_SCROLL_RANGE.

The search will always “wrap” the document from end to beginning or beginning to end. A previous or next find will never fail. If there is a single occurrence in the document, these will find the same string.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: OpenDoc, UndoFind

Example:

The following example performs a search.

C/C++ Example

```
CArsOle * pArsCtrl;  
CScrollBar * pHorzScrollBar, * pVertScrollBar;  
VARIANT found, horz_position, vert_position;  
char * pString;  
short rc;  
.  
.  
.  
  
rc = pArsCtrl->FindStringInDoc( pString,  
                                1,  
                                ARS_OLE_FIND_FIRST,  
                                FALSE,  
                                &found,  
                                &horz_position,  
                                &vert_position );  
  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
if ( found.iVal )  
{  
    pHorzScrollBar->SetScrollPos( (int)horz_position.iVal );  
    pVertScrollBar->SetScrollPos( (int)vert_position.iVal );  
    .  
    .  
}  
else  
{  
    .  
    .  
}  
  
.  
.
```

Visual Basic Example

```
Dim rc As Integer
Dim found, horz_pos, vert_pos As Variant
Dim Temp As String
.
>
.
rc = ArsOle.FindStringInDoc( Temp,
                             1,
                             ARS_OLE_FIND_FIRST,
                             False,
                             found,
                             horz_pos,
                             vert_pos )
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

If found <> 0 Then
    hScrollBar.Value = horz_pos
    vScrollBar.Value = vert_pos
End If

.
.
```

GetAnnotationForDoc

Note: This method intended for use with Visual Basic.

```
short GetAnnotationForDoc(  
    short      Index,  
    BSTR *     pText,  
    BSTR *     pUserId,  
    BSTR *     pDateTime,  
    VARIANT *  pPage,  
    VARIANT *  pPublic,  
    VARIANT *  pCanBeCopied )
```

Parameters: **Index**

Specifies the zero-based index of the annotation to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumDocAnnotations.

pText

Points to a BSTR to receive the text of the annotation.

pUserId

Points to a BSTR to receive the userid for the annotation.

pDateTime

Points to a BSTR to receive the date and time for the annotation.

pPage

Points to a variable to receive the document page number for the annotation. On return, this variable is set to type VT_I4.

pPublic

Points to a variable to receive a boolean flag indicating whether the annotation is public or private. On return, this variable is set to type VT_I2.

pCanBeCopied

Points to a variable to receive a boolean flag indicating whether the annotation can be copied to another server. On return, this variable is set to type VT_I2.

Description: The annotation is retrieved.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetNumDocAnnotations, GetDocAnnotation

Example

The following example retrieves an annotation for a document.

```
Dim rc, j As Integer
Dim num_notes, page, ispublic, canbecopied As Variant
Dim text As String
Dim userid As String
Dim datetime As String

rc = ArsOle.GetNumDocAnnotations( num_notes )
if rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For j = 0 To num_notes -1
    rc = ArsOle.GetAnnotationForDoc( j,
                                     text,
                                     userid,
                                     datetime,
                                     page,
                                     ispublic,
                                     canbecopied )

    if rc <> ARS_OLE_RC_SUCCESS Then
        MsgBox "ERROR"
    End
End If
Next j
```

GetAnnotationStatus

short **GetAnnotationStatus**(
 long **Index**,
 VARIANT * **pStatus**)

Parameters: **Index**

Specifies the zero-based index of a document within the document list of the active folder. If this value is less than zero, status is returned for the open document.

pStatus

Points to a variable to receive the annotation status. This will be one of the annotation status values, such as ARS_OLE_ANNOTATION_YES, found in ARSOLEEX.H. On return, this variable is set to type VT_I2.

Description: The annotation status is returned in the specified variable.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: AnnotateDoc, GetNumDocAnnotations, GetDocAnnotation, GetAnnotationForDoc

Example

The following example gets the annotation status for a document.

C/C++ Example

```
VARIANT status;
CArsOle * pArsCtrl;
short rc;

.
.
.

rc = pArsCtrl->GetAnnotationStatus( -1, &status );

if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

.
.
.
```

Visual Basic Example

```
Dim rc As Integer
Dim status As Variant

.
.
.

rc = ArsOle.GetAnnotationStatus( -1, status )
if rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.
.
```

GetControlId

short **GetControlId**(
 VARIANT * **pControlId**)

Parameters: **pControlId**

Points to a variable to receive the control id. On return, this variable is set to type VT_I4.

Description: The identifier of the control is returned in the specified variable. This control identifier can be used to reference information associated with a different OnDemand OLE Control. Refer to “Viewing Multiple Documents for a Single Folder” on page 3 for a discussion of control ids.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: OpenDoc

Example: The following example retrieves the control id.

C/C++ Example

```
long ControlId;  
  
.  
.  
  
CArsOle * pArsCtrl;  
VARIANT control_id;  
short rc;  
  
.  
.  
  
rc = pArsCtrl->GetControlId( &control_id );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
ControlId = control_id.lVal;  
  
.  
.
```

Visual Basic Example

```
Dim rc As Integer
Dim control_id As Variant

.
.

rc = ArsOle.GetControlId (control_id)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.
```

GetDocAnnotation

Note: This method intended for use with C/C++.

```
short GetDocAnnotation(  
    short      Index,  
    LPUNKNOWN pText,  
    LPUNKNOWN pUserId,  
    LPUNKNOWN pDateTime,  
    VARIANT *  pPage,  
    VARIANT *  pPublic,  
    VARIANT *  pCanBeCopied )
```

Parameters: **Index**

Specifies the zero-based index of the annotation to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumDocAnnotations.

pText

Points to a string to receive the text of the annotation.

pUserId

Points to a string to receive the userid for the annotation.

pDateTime

Points to a string to receive the date and time for the annotation.

pPage

Points to a variable to receive the document page number for the annotation. On return, this variable is set to type VT_I4.

pPublic

Points to a variable to receive a boolean flag indicating whether the annotation is public or private. On return, this variable is set to type VT_I2.

pCanBeCopied

Points to a variable to receive a boolean flag indicating whether the annotation can be copied to another server. On return, this variable is set to type VT_I2.

Description: The annotation data is retrieved.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetNumDocAnnotations, GetAnnotationForDoc

Example

The following example retrieves an annotation for a document.

```
VARIANT num_notes, page, ispublic, canbecopied;
CArsOle * pArsCtrl;
short rc, j;
char * pText, userid[100], datetime[200];

rc = pArsCtrl->GetNumDocAnnotations( &num_notes );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

pText = new char[35000];

for ( j = 0; j < num_notes.iVal; j++ )
{
    rc = pArsCtrl->GetDocAnnotation( j,
                                     (LPUNKNOWN)pText,
                                     (LPUNKNOWN)userid,
                                     (LPUNKNOWN)datetime,
                                     &page,
                                     &ispublic,
                                     &canbecopied );

    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;

    // Process annotation
}

delete pText;
```

GetDocBackgroundColor

```
short GetDocBackgroundColor(  
    VARIANT * pColor,  
    VARIANT * pChangeable )
```

Parameters: **pColor**

Points to a variable to receive the current document background color. This will be one of the color values, such as ARS_OLE_COLOR_WHITE, found in ARSOLEEX.H. On return, this variable is set to type VT_I2.

pChangeable

Points to a variable to receive an indication of whether the document background color can be changed. On return, this variable contains a non-zero value if the color is changeable; otherwise, zero. On return, this variable is set to type VT_I2.

Description: The current document background color and a changeability indicator are returned.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: SetDocBackgroundColor

Example: The following example retrieves the current document background color and disables a menu item if the color cannot be changed.

C/C++ Example

```
CArsOle * pArsCtrl;  
CMenu * pSubMenu;  
short rc, back_color;  
VARIANT current_color, changeable;  
  
:  
:  
  
rc = pArsCtrl->GetDocBackgroundColor( &current_color, &changeable );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
back_color = current_color.iVal;  
  
pSubMenu->EnableMenuItem(  
    ID_VIEW_BKGD_COLOR,  
    MF_BYCOMMAND | ( changeable.iVal ? MF_ENABLED : MF_GRAYED ) );  
  
:  
:
```


Visual Basic Example

```
Dim rc As Integer
Dim back_color, changeable As Variant

.
.

rc = ArsOle.GetDocBackgroundColor (back_color, changeable)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

If changeable <> 0 Then
    menuBackgroundColor.Enabled = True
Else
    menuBackgroundColor.Enabled = False
End If

.
.
```

GetDocCurrentPage

short **GetDocCurrentPage**(
 VARIANT * **pPage**)

Parameters: **pPage**

Points to a variable to receive the current page number of the open document. On return, this variable is set to type VT_I4.

Description: The current page number of the open document is returned in the specified variable.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: SetDocCurrentPage, GetDocNumPages

Example: The following example retrieves the current page number of the open document.

C/C++ Example

```
CArsOle * pArsCtrl;  
VARIANT vari;  
long page_num;  
short rc;  
  
.  
.  
  
rc = pArsCtrl->GetDocCurrentPage( &vari );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
page_num = var.lVal;  
  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
Dim page_num As Variant  
  
.  
.  
  
rc = ArsOle.GetDocCurrentPage (page_num)  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

GetDocDisplayValue

Note: This method intended for use with Visual Basic.

```
short GetDocDisplayValue(  
    long    DocIndex,  
    short   ValueIndex,  
    BSTR *  pValue )
```

Parameters: **DocIndex**

Specifies the zero-based index of a document within the document list of the active folder.

ValueIndex

Specifies the zero-based index of the value to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumFolderDisplayFields.

pValue

Points to a BSTR to receive the value.

Description: The specified value is returned in **pValue**.

GetDocDisplayValue or GetDocDisplayValues can be used to retrieve the document display values. An application should use the one which is more convenient in its environment.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetNumDocsInList, GetNumFolderDisplayFields, GetFolderDisplayFieldNames, GetDocDisplayValues, OpenDoc

Example:

The following example creates a listbox of the folder document list names and associated values and opens the document selected by a user.

```
Dim rc, count, i, j As Integer
Dim num_fields, num_docs As Variant
Dim Names() As String
Dim Line As String
Dim Temp As String
.
.
.
rc = ArsOle.GetNumFolderDisplayFields(num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

ReDim Names(num_fields)

For count = 0 To num_fields -1
    rc = ArsOle.GetFolderDisplayFieldName(count, Temp)
    Names(count) = Temp
Next count

rc = ArsOle.GetNumDocsInList(num_docs)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For j = 0 To num_docs -1
    For i = 0 To num_fields -1
        rc = ArsOle.GetDocDisplayValue(j, i, Temp)

        Line = Line + Names(i) + " = " + Temp
        If i < num_fields Then
            Line = Line + ", "
        End If
    Next i

    lbDocs.AddItem Line
Next j
.
.
.
'During OK button processing:

rc = ArsOle.OpenDoc (lbDocs.List(lbDocs.ListIndex), "", 0)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If
```

GetDocDisplayValues

Note: This method intended for use with C/C++.

```
short GetDocDisplayValues(  
    long      Index,  
    IUnknown * pValues,  
    short     MaxValues )
```

Parameters: **Index**

Specifies the zero-based index of a document within the document list of the active folder.

pValues

Points to an array of *ArsOleValues* to receive the values of the folder display fields for the document specified with **Index**. There are the same number of values as of display fields. The array must have at least **MaxValues** elements.

MaxValues

Specifies the maximum number of values to be returned.

Description:

The values of the folder display fields for the document, up to a maximum of **MaxValues**, are returned in **pValues**. Each name is a null-terminated character string.

The values are placed in the array in the same sequence that the display field names are returned by the *GetFolderDisplayFieldNames* method.

GetDocDisplayValue or *GetDocDisplayValues* can be used to retrieve the document display values. An application should use the one which is more convenient in its environment.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: *GetNumDocsInList*, *GetNumFolderDisplayFields*, *GetFolderDisplayFieldNames*, *GetDocDisplayValue*, *OpenDoc*

Example:

The following example creates a listbox of the folder document list names and associated values and opens the document selected by a user.

C/C++ example

```
CArsOle * pArsCtrl;
ArsOleName * pNames;
ArsOleValue * pValues;
CListBox * pDocList;
char * pLine;
short rc, k, opr, num_fields;
long j, num_docs;
int size;
VARIANT vari;
.
.
// During dialog initialization:

rc = pArsCtrl->GetNumFolderDisplayFields( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_fields = var.iVal;

pNames = new ArsOleName[ max( num_fields, 1 ) ];
rc = pArsCtrl->GetFolderDisplayFieldNames( (IUnknown*)pNames, num_fields );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

rc = pArsCtrl->GetNumDocsInList( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_docs = var.lVal;

pValues = new ArsOleValue[ max( num_fields, 1 ) ];

size = num_fields * ( sizeof(ArsOleName) + sizeof(ArsOleValue) + 5 );
pLine = new char[ size ];
for ( j = 0, pLine[0] = '\0'; j < num_docs; j++ )
{
    rc = pArsCtrl->GetDocDisplayValues( j, pValues, num_fields );
    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;
    .
    .
}
```

```

.
.
for ( k = 0; k < num_fields; k++ )
{
    strcat( pLine, pNames[k] );
    strcat( pLine, " = " );
    strcat( pLine, pValues[k] );
    if ( k < num_fields - 1 )
        strcat( pLine, ", " );
}
pDocList->InsertString( -1, pLine );
}
pDocList->SetCurSel( 0 );
.
.
// During OK button processing:

rc = pArsCtrl->OpenDoc( (long)pDocList->GetCurSel( ) , NULL, 0 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

.
.

```

GetDocImageColor

```
short GetDocImageColor(  
    VARIANT * pColor,  
    VARIANT * pChangeable )
```

Parameters: **pColor**

Points to a variable to receive the current document image color. This will be one of the color values, such as ARS_OLE_COLOR_BLACK, found in ARSOLEEX.H. On return, this variable is set to type VT_I2.

pChangeable

Points to a variable to receive an indication of whether the document image color can be changed. On return, this variable contains a non-zero value if the color is changeable; otherwise, zero. On return, this variable is set to type VT_I2.

Description: The current document image color and a changeability indicator are returned.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: SetDocImageColor

Example: The following example retrieves the current document image color and disables a menu item if the color cannot be changed.

C/C++ Example

```
CArsOle * pArsCtrl;  
CMenu * pSubMenu;  
short rc, image_color;  
VARIANT current_color, changeable;  
  
:  
:  
  
rc = pArsCtrl->GetDocImageColor( &current_color, &changeable );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
image_color = current_color.iVal;  
  
pSubMenu->EnableMenuItem(  
    ID_VIEW_IMAGE_COLOR,  
    MF_BYCOMMAND | ( changeable.iVal ? MF_ENABLED : MF_GRAYED ) );  
  
:  
:
```


Visual Basic Example

```
Dim rc As Integer
Dim current_color, changeable As Variant

.
.

rc = ArsOle.GetDocImageColor (current_color, changeable)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

If changeable <> 0 Then
    menuImageColor.Enabled = True
Else
    menuImageColor.Enabled = False
End If

.
.
```

GetDocImageIntensity

```
short GetDocImageIntensity(  
    VARIANT * pIntensity,  
    VARIANT * pChangeable )
```

Parameters: **pIntensity**

Points to a variable to receive the current document image intensity. This will be one of the intensity values, such as ARS_OLE_INTENSITY_NORMAL, found in ARSOLEEX.H. On return, this variable is set to type VT_I2.

pChangeable

Points to a variable to receive an indication of whether the document image intensity can be changed. On return, this variable contains a non-zero value if the intensity is changeable; otherwise, zero. On return, this variable is set to type VT_I2.

Description: The current document image intensity and a changeability indicator are returned.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: SetDocImageIntensity

Example: The following example retrieves the current document image intensity and disables a menu item if the intensity cannot be changed.

C/C++ Example

```
CArsOle * pArsCtrl;  
CMenu * pSubMenu;  
short rc, image_intensity;  
VARIANT current_intensity, changeable;  
  
:  
:  
  
rc = pArsCtrl->GetDocImageIntensity( &current_intensity, &changeable );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
image_intensity = current_intensity.iVal;  
  
pSubMenu->EnableMenuItem(  
    ID_VIEW_IMAGE_INTENSITY,  
    MF_BYCOMMAND | ( changeable.iVal ? MF_ENABLED : MF_GRAYED ) );  
  
:  
:
```

Visual Basic Example

```
Dim rc As Integer
Dim current_intensity, changeable As Variant

.
.

rc = ArsOle.GetDocImageIntensity (current_intensity, changeable)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

If changeable <> 0 Then
    menuImageIntensity.Enabled = True
Else
    menuImageIntensity.Enabled = False
End If

.
.
```

GetDocNumPages

short **GetDocNumPages**(
 VARIANT * **pNumPages**)

Parameters: **pNumPages**

Points to a variable to receive the number of pages in the open document. On return, this variable is set to type VT_I4.

Description: The number of pages in the open document is returned in the specified variable.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: OpenDoc, GetDocCurrentPage, SetDocCurrentPage

Example: The following example retrieves the number of pages in the open document.

C/C++ Example

```
CArsOle * pArsCtrl;  
VARIANT vari;  
long num_pages;  
short rc;  
  
.  
.  
  
rc = pArsCtrl->GetDocNumPages( &vari );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
num_pages = var.lVal;  
  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
Dim num_pages As Variant  
  
.  
.  
  
rc = ArsOle.GetDocNumPages (num_pages)  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

GetDocRotation

short **GetDocRotation**(
 VARIANT * **pRotation**,
 VARIANT * **pChangeable**)

Parameters: **pRotation**

Points to a variable to receive the current document rotation. This will be one of the rotation values, such as ARS_OLE_ROTATION_0, found in ARSOLEEX.H. On return, this variable is set to type VT_I2.

pChangeable

Points to a variable to receive an indication of whether the document rotation can be changed. On return, this variable contains a non-zero value if the rotation is changeable; otherwise, zero. On return, this variable is set to type VT_I2.

Description: The current document rotation and a changeability indicator are returned.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: SetDocRotation

Example: The following example retrieves the current document rotation and disables a menu item if the rotation cannot be changed.

C/C ++ Example

```
CArsOle * pArsCtrl;  
CMenu * pSubMenu;  
short rc, rotation;  
VARIANT current_rotation, changeable;  
  
.  
.  
  
rc = pArsCtrl->GetDocRotation( &current_rotation, &changeable );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
rotation = current_rotation.iVal;  
  
pSubMenu->EnableMenuItem(  
    ID_VIEW_ROTATION,  
    MF_BYCOMMAND | ( changeable.iVal ? MF_ENABLED : MF_GRAYED ) );  
.  
.
```

Visual Basic Example

```
Dim rc As Integer
Dim rotation, changeable As Variant

.
.

rc = ArsOle.GetDocRotation (rotation, changeable)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

If changeable <> 0 Then
    menuRotation.Enabled = True
Else
    menuRotation.Enabled = False
End If

.
.
```

GetDocScrollPositions

short **GetDocScrollPositions**(
 VARIANT * **pHorzPosition**,
 VARIANT * **pVertPosition**)

Parameters: **pHorzPosition**

Points to a variable to receive the new horizontal scroll position. On return, this variable is set to type VT_I2.

pVertPosition

Points to a variable to receive the new vertical scroll position. On return, this variable is set to type VT_I2.

Description: The current scroll positions are returned in the specified variables. The scroll positions assume that the scroll ranges have been set to ARS_OLE_SCROLL_RANGE.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also:

Example: The following example sets the current page number of the open document and updates the current scroll positions.

C/C++ Example

```
CArsOLE * pArsCtrl;  
CScrollBar * pHorzScrollBar, * pVertScrollBar;  
short rc;  
VARIANT horz_position, vert_position;  
.  
.  
rc = pArsCtrl->SetDocCurrentPage( 46 );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
.  
.  
  
rc = pArsCtrl->GetDocScrollPositions( &horz_position, &vert_position );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
pHorzScrollBar->SetScrollPos( (int)horz_position.iVal );  
pVertScrollBar->SetScrollPos( (int)vert_position.iVal );  
.  
.
```

Visual Basic Example

```
Dim rc As Integer
Dim horz_pos, vert_pos As Variant
```

```
.
.
```

```
rc = ArsOle.SetDocCurrentPage( 46 )
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If
```

```
rc = ArsOle.GetDocScrollPositions( horz_pos, vert_pos )
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If
```

```
sbHorz.Value = horz_pos
sbVert.Value = vert_pos
```

```
.
.
```

GetDocType

Note: This method intended for use with C/C++.

```
short GetDocType(  
    long          Index,  
    VARIANT *     pType,  
    LPUNKNOWN     pExtension )
```

Parameters: **Index**

Specifies the zero-based index of a document within the document list of the active folder. If this value is less than zero, then the open document is used.

pType

Points to a variable to receive the document type of the specified document. The document type will be one of the document type values found in ARSOLEEX.H, such as ARS_OLE_DOC_TYPE_AFP.

pExtension

Points to a string to receive the file extension of the document. This value is returned only if the document type is ARS_OLE_DOC_TYPE_USER_DEF.

Description: Retrieves the document type. If the document type is ARS_OLE_DOC_TYPE_USER_DEF, then the file extension is also retrieved.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: GetTypeForDoc

Example

The following example retrieves the document type for the third item in the document list.

```
VARIANT type;
char ext[ 20 ];
CArsOle * pArsCtrl;
short rc;

.
.
.

rc = pArsCtrl->GetDocType( 2, &type, extension );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

.
.
.
```

GetDocZoom

```
short GetDocZoom(  
    VARIANT * pCurrentZoomPercent,  
    VARIANT * pMinZoomPercent,  
    VARIANT * pMaxZoomPercent )
```

Parameters: **pCurrentZoomPercent**

Points to a variable to receive the current zoom percent. On return, this variable is set to type VT_I2.

pMinZoomPercent

Points to a variable to receive the minimum zoom percent. On return, this variable is set to type VT_I2.

pMaxZoomPercent

Points to a variable to receive the maximum zoom percent. On return, this variable is set to type VT_I2.

Description: The current, minimum, and maximum zoom percents for the document are returned in the specified variables.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: SetDocZoom

Example: The following example retrieves the current, minimum, and maximum zoom percents.

C/C++ Example

```
CArsOLE * pArsCtrl;  
short rc, current_zoom, min_zoom, max_zoom;  
VARIANT var1, var2, var3;  
  
.  
.  
  
rc = pArsCtrl->GetDocZoom( &var1, &var2, &var3 );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
current_zoom = var1.iVal;  
min_zoom = var2.iVal;  
max_zoom = var3.iVal;  
  
.  
.
```

Visual Basic Example

```
Dim rc As Integer
Dim current_zoom, min_zoom, max_zoom As Variant

.
.

rc = ArsOle.GetDocZoom (current_zoom, min_zoom, max_zoom)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.
```

GetFolderDisplayFieldName

Note: This method intended for use with Visual Basic.

```
short GetFolderDisplayFieldName(  
    short    Index,  
    BSTR *   pName )
```

Parameters: **Index**

Specifies the zero-based index of the name to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumFolderDisplayFields.

pName

Points to a BSTR to receive the name of the field.

Description: The specified field name is returned in **pName**.

GetFolderDisplayFieldName or GetFolderDisplayFieldNames can be used to retrieve the folder display field names. An application should use the one which is more convenient in its environment.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetNumFolderDisplayFields, GetFolderDisplayFieldNames

Example:

The following example creates a listbox of the folder document list names and associated values and opens the document selected by a user.

Visual Basic Example

```
Dim rc, count, i, j As Integer
Dim num_fields, num_docs As Variant
Dim Names() As String
Dim Line As String
Dim Temp As String
.
.
.
rc = ArsOle.GetNumFolderDisplayFields(num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
    End
End If

ReDim Names(num_fields)

For count = 0 To num_fields - 1
    rc = ArsOle.GetFolderDisplayFieldName(count, Temp)
    Names(count) = Temp
Next count

rc = ArsOle.GetNumDocsInList(num_docs)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
    End
End If
```

```

For j = 0 To num_docs -1
    For i = 0 To num_fields -1
        rc = ArsOle.GetDocDisplayValue(j, i, Temp)

        Line = Line + Names(i) + " = " + Temp
        If i < num_fields Then
            Line = Line + ", "
        End If
    Next i

    lbDocs.AddItem Line
Next j
.
.
.
'During OK button processing:

rc = ArsOle.OpenDoc (lbDocs.List(lbDocs.ListIndex), "", 0)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

GetFolderDisplayFieldNames

Note: This method intended for use with C/C++.

```
short GetFolderDisplayFieldNames(  
    IUnknown * pNames,  
    short      MaxNames )
```

Parameters: **pNames**

Points to an array of ArsOleNames to receive the names of the display fields for the active folder. The array must have at least **MaxNames** elements.

MaxNames

Specifies the maximum number of names to be returned.

Description: The names of the display fields for the active folder, up to a maximum of **MaxNames**, are returned in **pNames**. Each name is a null-terminated character string.

GetFolderDisplayFieldName or GetFolderDisplayFieldNames can be used to retrieve the folder display field names. An application should use the one which is more convenient in its environment.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetNumFolderDisplayFields, GetFolderDisplayFieldName

Example: The following example creates a listbox of the folder document list names and associated values and opens the document selected by a user.

C/C++ Example

```
CArsOle * pArsCtrl;  
ArsOleName * pNames;  
ArsOleValue * pValues;  
CListBox * pDocList;  
char * pLine;  
short rc, k, opr, num_fields;  
long j, num_docs;  
int size;  
VARIANT vari;  
.  
.  
// During dialog initialization:  
  
rc = pArsCtrl->GetNumFolderDisplayFields( &vari );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
num_fields = var.iVal;  
.  
.
```



```

.
.
pNames = new ArsOleName[ max( num_fields, 1 ) ];
rc = pArsCtrl->GetFolderDisplayFieldNames( (IUnknown*)pNames, num_fields );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

rc = pArsCtrl->GetNumDocsInList( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_docs = var.lVal;

pValues = new ArsOleValue[ max( num_fields, 1 ) ];

size = num_fields * ( sizeof(ArsOleName) + sizeof(ArsOleValue) + 5 );
pLine = new char[ size ];
for ( j = 0, pLine[0] = '\0'; j < num_docs; j++ )
{
    rc = pArsCtrl->GetDocDisplayValues( j, pValues, num_fields );
    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;

    for ( k = 0; k < num_fields; k++ )
    {
        strcat( pLine, pNames[k] );
        strcat( pLine, " = " );
        strcat( pLine, pValues[k] );
        if ( k < num_fields - 1 )
            strcat( pLine, ", " );
    }
    pDocList->InsertString( -1, pLine );
}
pDocList->SetCurSel( 0 );
.
.
// During OK button processing:

rc = pArsCtrl->OpenDoc( (long)pDocList->GetCurSel( ) , NULL, 0 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

GetFolderFieldName

Note: This method intended for use with Visual Basic.

```
short GetFolderFieldName(  
    short    Index,  
    BSTR *   pName )
```

Parameters: **Index**

Specifies the zero-based index of the value to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumFolderFields.

pName

Points to a BSTR to receive the field name.

Description: The specified value is returned in **pName**.

GetFolderFieldName or GetFolderFieldNames can be used to retrieve the folder field names. An application should use the one which is more convenient in its environment.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: GetFolderFieldNames, GetNumFolderFields, StoreDoc

Example: **Visual Basic Example**

```
Dim rc, count As Integer  
Dim num_fields As Variant  
Dim FieldNames() As String  
Dim Temp As String  
  
.  
.  
  
rc = ArsOle.GetNumFolderFields (num_fields)  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
ReDim FieldNames (num_fields - 1)  
  
For count = 0 To num_fields - 1  
    rc = ArsOle.GetFolderFieldName (count, Temp)  
    FieldNames(count) = Temp  
Next count  
  
.  
.
```

GetFolderFieldNames

Note: This method intended for use with C/C++.

```
short GetFolderFieldNames(  
    IUnknown * pNames,  
    short      MaxNames )
```

Parameters: **pNames**

Points to an array of ArsOleValues to receive the folder field names. The array must have at least MaxNames elements.

MaxNames

Specifies the maximum number of names to be returned.

Description:

The names of the folder fields, up to a maximum of MaxNames, are returned in pNames. Each name is a null-terminated character string.

The names are placed in the array in the same sequence that should be used with the method StoreDoc.

GetFolderFieldName or GetFolderFieldNames can be used to retrieve the folder field names. An application should use the one which is more convenient in its environment.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: GetFolderFieldName, GetNumFolderFields, StoreDoc

Example: The following example demonstrates the StoreDoc method. First the folder fields are displayed along with entry fields so that the user can enter field values. Then those values are used to store a new document into OnDemand.

C/C++ Example

```
VARIANT var;  
CArsOle * pArsCtrl;  
ArsOleName * pNames;  
short rc, j;  
  
.  
.  
  
rc = pArsCtrl->GetNumFolderFields( &var );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;
```

```

// m_NumFolderFields is a class variable
m_NumFolderFields = var.iVal;

pNames = new ArsOleName[ max( m_NumFolderFields, 1 ) ];
rc = pArsCtrl->GetFolderFieldNames( (IUnknown*)pNames,
                                     m_NumFolderFields );

if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
for ( j = 0; j < m_NumFolderFields; j++ )
    GetDlgItem( IDC_FIELD1_TEXT + j )->SetWindowText( pNames[j] );

// During OK button processing

CArsOle * pArsCtrl;
short rc, j;
CString fields[16];
SAFEARRAY * pSA;
VARIANT var;
BSTR bstrElement;
long i;

pSA = SafeArrayCreateVector(VT_BSTR, 0, m_NumFolderFields);
if ( pSA == NULL )
    ERROR;

for (j = 0; j < m_NumFolderFields; j++)
    GetDlgItem( IDC_FIELD1_EDIT + j )->GetWindowText( fields[j] );

for (i = 0; i < m_NumFolderFields; i++)
{
    bstrElement = fields[i].AllocSysString();
    if (bstrElement == NULL)
        ERROR;
    SafeArrayPutElement (pSA, &i, bstrElement);
}

var.vt = VT_ARRAY | VT_BSTR;
var.parray = pSA;

rc = pArsCtrl->StoreDoc( "G:\\download\\file.afp",
                        "BKH-CRD",
                        "BKH-CRD",
                        &var );

if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

```

GetFolderName

Note: This method intended for use with Visual Basic.

```
short GetFolderName(  
    short    Index,  
    BSTR *   pName )
```

Parameters: **Index**

Specifies the zero-based index of the name to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumFolders.

pName

Points to a BSTR to receive the name of the folder.

Description: The specified folder name is returned in **pName**.

GetFolderName or GetFolderNames can be used to retrieve the folder names. An application should use the one which is more convenient in its environment.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: GetNumFolders, GetFolderNames, OpenFolder

Example:

Visual Basic Example

```

Dim rc, count As Integer
Dim num_folders As Variant
Dim Temp As String

.
.

rc = ArsOle.GetNumFolders (num_folders)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For count = 0 To num_folders -1
    rc = ArsOle.GetFolderName(count, Temp)
    lbFolders.AddItem Temp
Next count

.
.

' During OK button processing

rc = ArsOle.OpenFolder (lbFolders.List(lbFolders.ListItem))
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

GetFolderNames

Note: This method intended for use with C/C++.

```
short GetFolderNames(  
    IUnknown * pNames,  
    short      MaxNames )
```

Parameters: **pNames**

Points to an array of **ArsOleNames** to receive the names of the folders available for the current server. The array must have at least **MaxNames** elements.

MaxNames

Specifies the maximum number of names to be returned.

Description: The names of the folders available for the current server, up to a maximum of **MaxNames**, are returned in **pNames**. Each name is a null-terminated character string.

GetFolderName or GetFolderNames can be used to retrieve the folder names. An application should use the one which is more convenient in its environment.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetNumFolders, GetFolderName, OpenFolder

Example:

The following example retrieves the names of all folders available for the current server, puts them in a ComboBox control, retrieves the chosen folder, and performs an open for that folder.

C/C++ Example

```
CArsOle * pArsCtrl;
ArsOleName * pFolderNames;
CComboBox * pFoldersList;
char folder[ sizeof( ArsOleName ) ];
short rc, j, num_folders;
int index;
VARIANT vari;
.
.

// During dialog initialization:

rc = pArsCtrl->GetNumFolders( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_folders = var.iVal;

pFolderNames = new ArsOleName[ max( num_folders, 1 ) ];
rc = pArsCtrl->GetFolderNames( (IUnknown*)pFolderNames, num_folders );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < num_folders; j++ )
    index = pFoldersList->AddString( pFolderNames[j] );
pFoldersList->SetCurSel( 0 );
.
.

// During OK button processing:

pFoldersList->GetWindowText( folder, sizeof(folder) );

rc = pArsCtrl->OpenFolder( folder );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.
```

GetFolderSearchFieldName

Note: This method intended for use with Visual Basic.

```
short GetFolderSearchFieldName(  
    short    Index,  
    BSTR *   pName )
```

Parameters: **Index**

Specifies the zero-based index of the name to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumFolderSearchFields.

pName

Points to a BSTR to receive the name of the field.

Description: The specified field name is returned in **pName**.

GetFolderSearchFieldName or GetFolderSearchFieldNames can be used to retrieve the folder search field names. An application should use the one which is more convenient in its environment.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetNumFolderSearchFields, GetFolderSearchFieldNames, SetFolderSearchFieldData, SearchFolder

Example:

Visual Basic Example

```
Dim rc, count, i, j As Integer
Dim num_fields, num_docs As Variant
Dim Names() As String
Dim Line As String
Dim Temp As String
Dim Oprs As Variant
.
.
.
Oprs = Array ("Equal", "Not Equal", ..., "Like", "Not Like")

rc = ArsOle.GetNumFolderSearchFields(num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

ReDim Names(num_fields -1)

For count = 0 To num_fields -1
    rc = ArsOle.GetFolderSearchFieldName(count, Temp)
    Names(count) = Temp
Next count

for count = 0 To num_fields -1
    lbFieldList.AddItem Names(count)
Next count

for count = 0 To UBound(Oprs)
    lbOprList.AddItem (Oprs(count))
Next count
.
.
.
```

```

.
.
.
' During SET FIELD button processing
rc = ArsOLE.SetFolderSearchFieldData (lbFieldList.List(lbFieldList.ListIndex),
                                      lbOprList.ListIndex,
                                      txtValue1.Value,
                                      txtValue2.Value)

If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

'During OK button processing:

rc = ArsOLE.SearchFolder (False)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

GetFolderSearchFieldNames

Note: This method intended for use with C/C++.

```
short GetFolderSearchFieldNames(  
    IUnknown * pNames,  
    short      MaxNames )
```

Parameters: **pNames**

Points to an array of ArsOleNames to receive the names of the search fields for the active folder. The array must have at least **MaxNames** elements.

MaxNames

Specifies the maximum number of names to be returned.

Description: The names of the search fields for the active folder, up to a maximum of **MaxNames**, are returned in **pNames**. Each name is a null-terminated character string.

GetFolderSearchFieldName or GetFolderSearchFieldNames can be used to retrieve the folder search field names. An application should use the one which is more convenient in its environment.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetNumFolderSearchFields, GetFolderSearchFieldName, SetFolderSearchFieldData, SearchFolder

Example: The following example retrieves the names of the active folder search fields, gives a user the opportunity to set the values for these fields, and initiates a search of the folder.

C/C++ Example

```
CArsOle * pArsCtrl;  
ArsOleName * pNames;  
CListBox * pFieldList, * pOprList;  
CEdit * pValue1, * pValue2;  
char name[ sizeof( ArsOleName ) ];  
char value1[ sizeof( ArsOleValue ) ];  
char value2[ sizeof( ArsOleValue ) ];  
short rc, j, opr, num_fields;  
VARIANT vari;  
.  
.  
struct _OprMap  
{  
    short code;  
    char * pText;  
} OprMap  
.  
.
```

```

.
.
static OprMap Oprs[] =
{ { ARS_OLE_OPR_EQUAL,          "Equal" },
  { ARS_OLE_OPR_NOT_EQUAL,      "Not Equal" },
  .
  .
  { ARS_OLE_OPR_LIKE,           "Like" },
  { ARS_OLE_OPR_NOT_LIKE,       "Not Like" } };

#define NUM_OPRS ( sizeof(Oprs) / sizeof(OprMap) )

// During dialog initialization:

rc = pArsCtrl->GetNumFolderSearchFields( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_fields = var.iVal;

pNames = new ArsOleName[ max( num_fields, 1 ) ];
rc = pArsCtrl->GetFolderSearchFieldNames( (IUnknown*)pNames, num_fields );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < num_fields; j++ )
    pFieldList->InsertString( -1, pNames[j] );
pFieldList->SetCurSel( 0 );

for ( j = 0; j < NUM_OPRS; j++ )
{
    pOprList->InsertString( -1, Oprs[j].pText );
    pOprList->SetItemData( j, (DWORD)Oprs[j].code );
}
pOprList->SetCurSel( 0 );
.
.

```

```

.
.
// During SET FIELD button processing:

pFieldList->GetText( pFieldList->GetCurSel( ), name );
opr = (short)pOprList->GetItemData( pOprList->GetCurSel( ) );
pValue1->GetWindowText( value1, sizeof(value1) );
pValue2->GetWindowText( value2, sizeof(value2) );

rc = pArsCtrl->SetFolderSearchFieldData( name, opr, value1, value2 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.
// During OK button processing:

rc = pArsCtrl->SearchFolder( FALSE );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

GetNumDocAnnotations

short **GetNumDocAnnotations**(
 VARIANT * **pNumAnnotations**)

Parameters: **pNumAnnotations**

Points to a variable to receive the number of annotations attached to the document.
On return, this variable is set to type VT_I4.

Description: The number of annotations attached to the document is returned in the specified variable.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: GetDocAnnotation, GetAnnotationForDoc

Example: The following example retrieves the annotations for a document.

C/C++ Example

```
VARIANT num_notes, page, ispublic, canbecopied;  
CArsOle * pArsCtrl;  
short rc, j;  
char * pText, userid[100], datetime[200];  
  
rc = pArsCtrl->GetNumDocAnnotations( &num_notes );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
pText = new char[35000];  
  
for ( j = 0; j < num_notes.iVal; j++ )  
{  
    rc = pArsCtrl->GetDocAnnotation( j,  
                                    (LPUNKNOWN)pText,  
                                    (LPUNKNOWN)userid,  
                                    (LPUNKNOWN)datetime,  
                                    &page,  
                                    &ispublic,  
                                    &canbecopied );  
  
    if ( rc != ARS_OLE_RC_SUCCESS )  
        ERROR;  
  
    // Process annotation  
}  
  
delete pText;
```

Visual Basic Example

```
Dim rc, j As Integer
Dim num_notes, page, ispublic, canbecopied As Variant
Dim text As String
Dim userid As String
Dim datetime As String

rc = ArsOle.GetNumDocAnnotations( num_notes );
if rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For j = 0 To num_notes -1
    rc = ArsOle.GetAnnotationForDoc( j,
                                     text,
                                     userid,
                                     datetime,
                                     page,
                                     ispublic,
                                     canbecopied )

    if rc <> ARS_OLE_RC_SUCCESS Then
        MsgBox "ERROR"
    End
End If

    ' Process Annotation

Next j
```

GetNumDocsInList

short **GetNumDocsInList**(
 VARIANT * **pNumDocs**)

Parameters: **pNumDocs**

Points to a variable to receive the number of documents in the document list of the active folder. On return, this variable is set to type VT_I4.

Description: The number of documents in the document list of the active folder is returned in the specified variable.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: GetDocDisplayValues, OpenDoc

Example: The following example creates a listbox of the folder document list names and associated values and opens the document selected by a user.

C/C++ Example

```
CArsOle * pArsCtrl;  
ArsOleName * pNames;  
ArsOleValue * pValues;  
CListBox * pDocList;  
char * pLine;  
short rc, k, opr, num_fields;  
long j, num_docs;  
int size;  
VARIANT vari;  
.  
.  
// During dialog initialization:  
  
rc = pArsCtrl->GetNumFolderDisplayFields( &vari );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
num_fields = var.iVal;  
  
pNames = new ArsOleName[ max( num_fields, 1 ) ];  
rc = pArsCtrl->GetFolderDisplayFieldNames( (IUnknown*)pNames, num_fields );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
rc = pArsCtrl->GetNumDocsInList( &vari );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
num_docs = var.lVal;  
  
pValues = new ArsOleValue[ max( num_fields, 1 ) ];  
.  
.
```

```

    .
    .
size = num_fields * ( sizeof(ArsOleName) + sizeof(ArsOleValue) + 5 );
pLine = new char[ size ];
for ( j = 0, pLine[0] = '\\0'; j < num_docs; j++ )
{
    rc = pArsCtrl->GetDocDisplayValues( j, pValues, num_fields );
    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;

    for ( k = 0; k < num_fields; k++ )
    {
        strcat( pLine, pNames[k] );
        strcat( pLine, " = " );
        strcat( pLine, pValues[k] );
        if ( k < num_fields - 1 )
            strcat( pLine, ", " );
    }
    pDocList->InsertString( -1, pLine );
}
pDocList->SetCurSel( 0 );
    .
    .
// During OK button processing:

rc = pArsCtrl->OpenDoc( (long)pDocList->GetCurSel( ) , NULL, 0 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
    .
    .

```

Visual Basic Example

```
Dim rc, count, i, j As Integer
Dim num_fields, num_docs As Variant
Dim Names() As String
Dim Line As String
Dim Temp As String

.
.
.
rc = ArsOle.GetNumFolderDisplayFields(num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

ReDim Names(num_fields - 1)

For count = 0 To num_fields - 1
    rc = ArsOle.GetFolderDisplayName(count, Temp)
    Names(count) = Temp
Next count

rc = ArsOle.GetNumDocsInList(num_docs)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For j = 0 To num_docs - 1
    For i = 0 To num_fields - 1
        rc = ArsOle.GetDocDisplayValue(j, i, Temp)

        Line = Line + Names(i) + " = " + Temp
        If i < num_fields Then
            Line = Line + ", "
        End If
    Next i

    lbDocs.AddItem Line
Next j

.
.
.
'During OK button processing:

rc = ArsOle.OpenDoc (lbDocs.List(lbDocs.ListIndex), "", 0)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If
```

GetNumFolderDisplayFields

short **GetNumFolderDisplayFields**(
 VARIANT * **pNumFields**)

Parameters: **pNumFields**

Points to a variable to receive the number of display fields for the active folder. On return, this variable is set to type VT_I2.

Description: The number of display fields for the active folder is returned in the specified variable.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: GetFolderDisplayFieldNames

Example: The following example creates a listbox of the folder document list names and associated values and opens the document selected by a user.

C/C++ Example

```
CArsOle * pArsCtrl;  
ArsOleName * pNames;  
ArsOleValue * pValues;  
CListBox * pDocList;  
char * pLine;  
short rc, k, opr, num_fields;  
long j, num_docs;  
int size;  
VARIANT vari;  
.  
.  
// During dialog initialization:  
  
rc = pArsCtrl->GetNumFolderDisplayFields( &vari );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
num_fields = var.iVal;  
  
pNames = new ArsOleName[ max( num_fields, 1 ) ];  
rc = pArsCtrl->GetFolderDisplayFieldNames( (IUnknown*)pNames, num_fields );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
rc = pArsCtrl->GetNumDocsInList( &vari );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
num_docs = var.lVal;  
.  
.
```

```

.
.
pValues = new ArsOleValue[ max( num_fields, 1 ) ];
size = num_fields * ( sizeof(ArsOleName) + sizeof(ArsOleValue) + 5 );
pLine = new char[ size ];
for ( j = 0, pLine[0] = '\0'; j < num_docs; j++ )
{
    rc = pArsCtrl->GetDocDisplayValues( j, pValues, num_fields );
    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;

    for ( k = 0; k < num_fields; k++ )
    {
        strcat( pLine, pNames[k] );
        strcat( pLine, " = " );
        strcat( pLine, pValues[k] );
        if ( k < num_fields - 1 )
            strcat( pLine, ", " );
    }
    pDocList->InsertString( -1, pLine );
}
pDocList->SetCurSel( 0 );
.
.
// During OK button processing:

rc = pArsCtrl->OpenDoc( (long)pDocList->GetCurSel( ) , NULL, 0 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```

Dim rc, count, i, j As Integer
Dim num_fields, num_docs As Variant
Dim Names() As String
Dim Line As String
Dim Temp As String
.
.
.
rc = ArsOle.GetNumFolderDisplayFields(num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

ReDim Names(num_fields -1)

For count = 0 To num_fields -1
    rc = ArsOle.GetFolderDisplayFieldName(count, Temp)
    Names(count) = Temp
Next count

rc = ArsOle.GetNumDocsInList(num_docs)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For j = 0 To num_docs -1
    For i = 0 To num_fields -1
        rc = ArsOle.GetDocDisplayValue(j, i, Temp)

        Line = Line + Names(i) + " = " + Temp
        If i < num_fields Then
            Line = Line + ", "
        End If
    Next i

    lbDocs.AddItem Line
Next j
.
.
.
'During OK button processing:

rc = ArsOle.OpenDoc (lbDocs.List(lbDocs.ListIndex), "", 0)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

GetNumFolderFields

short **GetNumFolderFields**(
 VARIANT * **pNumfields**)

Parameters: **pNumfields**

Points to a variable to receive the number of folder fields for the active folder. On return, this variable is set to type VT_I2.

Description: The number of folder fields for the active folder is returned in the specified variable.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: GetFolderFieldName, GetFolderFieldNames

Example: The following example demonstrates the StoreDoc method. First the folder fields are displayed along with entry fields so that the user can enter field values. Then those values are used to store a new document into OnDemand.

C/C++ Example

```
VARIANT var;
CArsOle * pArsCtrl;
ArsOleName * pNames;
short rc, j;
.
.
rc = pArsCtrl->GetNumFolderFields( &var );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

// m_NumFolderFields is a class variable
m_NumFolderFields = var.iVal;

pNames = new ArsOleName[ max( m_NumFolderFields, 1 ) ];
rc = pArsCtrl->GetFolderFieldNames( (IUnknown*)pNames,
    m_NumFolderFields );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < m_NumFolderFields; j++ )
    GetDlgItem( IDC_FIELD1_TEXT + j )->SetWindowText( pNames[j] );

// During OK button processing
CArsOle * pArsCtrl;
short rc, j;
CString fields[16];
SAFEARRAY * pSA;
VARIANT var;
BSTR bstrElement;
long i;
```



```

pSA = SafeArrayCreateVector(VT_BSTR, 0, m_NumFolderFields);
if ( pSA == NULL )
    ERROR;

for (j = 0; j < m_NumFolderFields; j++)
    GetDlgItem( IDC_FIELD1_EDIT + j )->GetWindowText( fields[j] );

for (i = 0; i < m_NumFolderFields; i++)
{
    bstrElement = fields[i].AllocSysString();
    if (bstrElement == NULL)
        ERROR;
    SafeArrayPutElement (pSA, &i, bstrElement);
}

var.vt = VT_ARRAY | VT_BSTR;
var.parray = pSA;

rc = pArsCtrl->StoreDoc( "G:\\download\\file.afp",
                        "BKH-CRD",
                        "BKH-CRD",
                        &var );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

```

Visual Basic Example

```
Dim rc, count As Integer
Dim num_fields As Variant
Dim FieldNames() As String
Dim Temp As String
.
.

rc = ArsOle.GetNumFolderFields (num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

ReDim FieldNames (num_fields - 1)

For count = 0 To num_fields - 1
    rc = ArsOle.GetFolderFieldName (count, Temp)
    FieldNames(count) = Temp
Next count

.
.
```

GetNumFolders

short **GetNumFolders**(
 VARIANT * **pNumFolders**)

Parameters: **pNumFolders**

Points to a variable to receive the number of folders available for the current server. On return, this variable is set to type VT_I2.

Description: The number of folders available for the current server is returned in the specified variable. This value can be used with the GetFolderNames method to prepare for opening a folder.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: GetFolderNames, OpenFolder

Example:

The following example retrieves the names of all folders available for the current server, puts them in a ComboBox control, retrieves the chosen folder, and performs an open for that folder.

C/C++ Example

```
CArsOLE * pArsCtrl;
ArsOLEName * pFolderNames;
CComboBox * pFoldersList;
char folder[ sizeof( ArsOLEName ) ];
short rc, j, num_folders;
int index;
VARIANT vari;
.
.
// During dialog initialization:

rc = pArsCtrl->GetNumFolders( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_folders = var.iVal;

pFolderNames = new ArsOLEName[ max( num_folders, 1 ) ];
rc = pArsCtrl->GetFolderNames( (IUnknown*)pFolderNames, num_folders );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < num_folders; j++ )
    index = pFoldersList->AddString( pFolderNames[j] );
pFoldersList->SetCurSel( 0 );
.
.

// During OK button processing:

pFoldersList->GetWindowText( folder, sizeof(folder) );

rc = pArsCtrl->OpenFolder( folder );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.
```

Visual Basic Example

```
Dim rc, count As Integer
Dim num_folders As Variant
Dim Temp As String

.
.

rc = ArsOle.GetNumFolders (num_folders)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

ReDim FolderNames(num_folders -1)

For count = 0 To num_folders -1
    rc = ArsOle.GetFolderName(count, Temp)
    lbFolders.AddItem Temp
Next count

.
.

' During OK button processing

rc = ArsOle.OpenFolder (lbFolders.List(lbFolders.ListItem))
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If
```

GetNumFolderSearchFields

short **GetNumFolderSearchFields**(
 VARIANT * **pNumFields**)

Parameters: **pNumFields**

Points to a variable to receive the number of search fields for the active folder. On return, this variable is set to type VT_I2.

Description: The number of search fields for the active folder is returned in the specified variable. This value can be used with the GetFolderSearchFieldNames method to prepare for setting the search field values for a folder.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetFolderSearchFieldNames, SetFolderSearchFieldData, SearchFolder

Example: The following example retrieves the names of the active folder search fields, gives a user the opportunity to set the values for these fields, and initiates a search of the folder.

C/C++ Example

```
CArsOle * pArsCtrl;  
ArsOleName * pNames;  
CListBox * pFieldList, * pOprList;  
CEdit * pValue1, * pValue2;  
char name[ sizeof( ArsOleName ) ];  
char value1[ sizeof( ArsOleValue ) ];  
char value2[ sizeof( ArsOleValue ) ];  
short rc, j, opr, num_fields;  
VARIANT vari;  
.  
.  
struct _OprMap  
{  
    short code;  
    char * pText;  
} OprMap  
  
static OprMap Oprs[] =  
{ { ARS_OLE_OPR_EQUAL,                "Equal" },  
  { ARS_OLE_OPR_NOT_EQUAL,            "Not Equal" },  
  .  
  .  
  { ARS_OLE_OPR_LIKE,                 "Like" },  
  { ARS_OLE_OPR_NOT_LIKE,             "Not Like" } };  
  
#define NUM_OPRS ( sizeof(Oprs) / sizeof(OprMap) )  
.  
.
```

```

// During dialog initialization:

rc = pArsCtrl->GetNumFolderSearchFields( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_fields = var.iVal;

pNames = new ArsOleName[ max( num_fields, 1 ) ];
rc = pArsCtrl->GetFolderSearchFieldNames( (IUnknown*)pNames, num_fields );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < num_fields; j++ )
    pFieldList->InsertString( -1, pNames[j] );
pFieldList->SetCurSel( 0 );

for ( j = 0; j < NUM_OPRS; j++ )
{
    pOprList->InsertString( -1, Oprs[j].pText );
    pOprList->SetItemData( j, (DWORD)Oprs[j].code );
}
pOprList->SetCurSel( 0 );
.
.

// During SET FIELD button processing:

pFieldList->GetText( pFieldList->GetCurSel( ), name );
opr = (short)pOprList->GetItemData( pOprList->GetCurSel( ) );
pValue1->GetWindowText( value1, sizeof(value1) );
pValue2->GetWindowText( value2, sizeof(value2) );

rc = pArsCtrl->SetFolderSearchFieldData( name, opr, value1, value2 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

// During OK button processing:

rc = pArsCtrl->SearchFolder( FALSE );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```
Dim rc, count, i, j As Integer
Dim num_fields, num_docs As Variant
Dim Names() As String
Dim Line As String
Dim Temp As String
Dim Oprs As Variant
.
.
.
Oprs = Array ("Equal", "Not Equal", ..., "Like", "Not Like")

rc = ArsOle.GetNumFolderSearchFields(num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

ReDim Names(num_fields - 1)

For count = 0 To num_fields - 1
    rc = ArsOle.GetFolderSearchFieldName(count, Temp)
    Names(count) = Temp
Next count

for count = 0 To num_fields - 1
    lbFieldList.AddItem Names(count)
Next count

for count = 0 To UBound(Oprs)
    lbOprList.AddItem (Oprs(count))
Next count
.
.
.
' During SET FIELD button processing
rc = ArsOle.SetFolderSearchFieldData (lbFieldList.List(lbFieldList.ListIndex),
                                      lbOprList.ListIndex,
                                      txtValue1.Value,
                                      txtValue2.Value)

If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

'During OK button processing:

rc = ArsOle.SearchFolder (False)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If
```

GetNumServerPrinters

short **GetNumServerPrinters**(
 VARIANT * **pNumServerPrinters**)

Parameters: **pNumServerPrinters**

Points to a variable to receive the number of server printers available for the current server. On return, this variable is set to type VT_I2.

Description: The number of server printers available is returned in the specified variable. This value can be used with the GetServerPrinter and GetServerPrinterInfo methods.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: GetServerPrinter, GetServerPrinterInfo

Example:

The following example retrieves the names and attributes of the available server printers.

C/C++ Example

```
CArsOle * pArsCtrl;  
  
short rc, j, num_prts;  
char name[ 200 ];  
VARIANT vari, type;  
.  
.  
.  
  
rc = pArsCtrl->GetNumServerPrinters( &vari );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
num_prts = vari.iVal;  
  
for ( j = 0; j < num_prts; j++ )  
{  
    rc = pArsCtrl->GetServerPrinter( j, name, type );  
    if ( rc != ARS_OLE_RC_SUCCESS )  
        ERROR;  
  
    // Process name and type  
  
}
```

Visual Basic Example

```

Dim rc, count As Integer
Dim num_prts, type As Variant
Dim name As String

.
.

rc = ArsOle.GetNumServers (num_prts)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For count = 0 To num_prts -1
    rc = ArsOle.GetServerPrinterInfo (count, name, type)
    ' Process name and type
Next count

.
.

```

GetNumServers

short **GetNumServers**(
 VARIANT * **pNumServers**)

Parameters: **pNumServers**

Points to a variable to receive the number of servers available for logon. On return, this variable is set to type VT_I2.

Description: The number of servers available for logon is returned in the specified variable. This value can be used with the GetServerNames method to prepare for a logon.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: GetServerNames, Logon

Example:

The following example retrieves the names of all servers available for logon, puts them in a ComboBox control, retrieves the chosen server, userid, and password, and performs a logon.

C/C++ Example

```
CArsOle * pArsCtrl;
ArsOleName * pServerNames;
CComboBox * pServersList;
CEdit * pUserId;
CEdit * pPassword;
char server[ sizeof( ArsOleName ) ];
char user[ sizeof( ArsOleName ) ];
char password[ sizeof( ArsOleName ) ];
short rc, j, num_servers;
int index;
VARIANT vari;
.
.
// During dialog initialization:

rc = pArsCtrl->GetNumServers( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_servers = vari.iVal;

pServerNames = new ArsOleName[ max( num_servers, 1 ) ];
rc = pArsCtrl->GetServerNames( (IUnknown*)pServerNames, num_servers );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < num_servers; j++ )
    index = pServersList->AddString( pServerNames[j] );
pServersList->SetCurSel( 0 );
.
.
// During OK button processing:

pServersList->GetWindowText( server, sizeof(server) );
pUserId->GetWindowText( user, sizeof(user) );
pPassword->GetWindowText( password, sizeof(password) );

rc = pArsCtrl->Logon( server, user, password );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.
```

Visual Basic Example

```
Dim rc, count As Integer
Dim num_servers As Variant
Dim Temp As String

.
.

rc = ArsOle.GetNumServers (num_servers)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For count = 0 To num_servers -1
    rc = ArsOle.GetServerName(count, Temp)
    lbServers.AddItem Temp
Next count

.
.

' During OK button processing

rc = ArsOle.Logon (lbServers.List(lbServers.ListItem), txtUser.Value, txtPasswd.Value)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If
```

GetServerName

Note: This method intended for use with Visual Basic.

```
short GetServerName(  
    short    Index,  
    BSTR *   pName )
```

Parameters: **Index**

Specifies the zero-based index of the name to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumServers.

pName

Points to a BSTR to receive the name of the server.

Description: The specified server name is returned in **pName**.

GetServerName or GetServerNames can be used to retrieve the server names. An application should use the one which is more convenient in its environment.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: GetNumServers, GetServerNames, Logon

Example:

Visual Basic Example

```

Dim rc, count As Integer
Dim num_servers As Variant
Dim Temp As String

.
.

rc = ArsOle.GetNumServers (num_servers)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For count = 0 To num_servers -1
    rc = ArsOle.GetServerName(count, Temp)
    lbServers.AddItem Temp
Next count

.
.

' During OK button processing

rc = ArsOle.Logon (lbServers.List(lbServers.ListItem), txtUser.Value, txtPasswd.Value)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

GetServerNames

Note: This method intended for use with C/C++.

```
short GetServerNames(  
    IUnknown * pNames,  
    short      MaxNames )
```

Parameters: **pNames**

Points to an array of **ArsOleNames** to receive the names of the servers available for logon. The array must have at least **MaxNames** elements.

MaxNames

Specifies the maximum number of names to be returned.

Description: The names of the servers available for logon, up to a maximum of **MaxNames**, are returned in **pNames**. Each name is a null-terminated character string.

GetServerName or GetServerNames can be used to retrieve the server names. An application should use the one which is more convenient in its environment.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetNumServers, GetServerName, Logon

Example:

The following example retrieves the names of all servers available for logon, puts them in in a ComboBox control, retrieves the chosen server, userid, and password, and performs a logon.

C/C++ Example

```
CArsOle * pArsCtrl;
ArsOleName * pServerNames;
CComboBox * pServersList;
CEdit * pUserId;
CEdit * pPassword;
char server[ sizeof( ArsOleName ) ];
char user[ sizeof( ArsOleName ) ];
char password[ sizeof( ArsOleName ) ];
short rc, j, num_servers;
int index;
VARIANT vari;
.
.
// During dialog initialization:

rc = pArsCtrl->GetNumServers( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_servers = var.iVal;

pServerNames = new ArsOleName[ max( num_servers, 1 ) ];
rc = pArsCtrl->GetServerNames( (IUnknown*)pServerNames, num_servers );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < num_servers; j++ )
    index = pServersList->AddString( pServerNames[j] );
pServersList->SetCurSel( 0 );
.
.

// During OK button processing:

pServersList->GetWindowText( server, sizeof(server) );
pUserId->GetWindowText( user, sizeof(user) );
pPassword->GetWindowText( password, sizeof(password) );

rc = pArsCtrl->Logon( server, user, password );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.
```

GetServerPrinter

Note: This method intended for use with C/C++.

```
short GetServerPrinter(  
    short      Index,  
    LPUNKNOWN  pName,  
    VARIANT *   pType )
```

Parameters: **Index**

Specifies the zero-based index of the printer to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumServerPrinters.

pName

Points to a string to receive the name of the server printer.

pType

Points to a variable to receive the type of the server printer. It will be one of the following type values found in ARSOLEEX.H:

```
ARS_OLE_SERVER_PRINTER_PRINT  
ARS_OLE_SERVER_PRINTER_PRINT_WITH_INFO  
ARS_OLE_SERVER_PRINTER_FAX
```

On return, this variable is set to type VT_I2.

Description: The server printer information is retrieved.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: GetNumServerPrinters, GetServerPrinterInfo

Example

The following example retrieves the names and attributes of the available server printers.

```
CArsOle * pArsCtrl;

short rc, j, num_prts;
char name[ 200 ];
VARIANT vari, type;
.
.
.

rc = pArsCtrl->GetNumServerPrinters( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_prts = vari.iVal;

for ( j = 0; j < num_prts; j++ )
{
    rc = pArsCtrl->GetServerPrinter( j, name, type );
    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;

    // Process name and type
}
.
.
.
```

GetServerPrinterInfo

Note: This method intended for use with Visual Basic.

```
short GetServerPrinterInfo(  
    short          Index,  
    BSTR *         pName,  
    VARIANT *      pType )
```

Parameters: **Index**

Specifies the zero-based index of the printer to be returned. It must be a number greater than or equal to zero and less than the value returned by GetNumServerPrinters.

pName

Points to a BSTR to receive the name of the server printer.

pType

Points to a variable to receive the type of the server printer. It will be one of the following type values found in ARSOLEEX.H:

```
ARS_OLE_SERVER_PRINTER_PRINT  
ARS_OLE_SERVER_PRINTER_PRINT_WITH_INFO  
ARS_OLE_SERVER_PRINTER_FAX
```

Description: The server printer information is retrieved.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: GetNumServerPrinters, GetServerPrinter

Example

The following example retrieves the names and attributes of the available server printers.

```
Dim rc, count As Integer
Dim num_prts, type As Variant
Dim name As String

.
.

rc = ArsOle.GetNumServers (num_prts)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For count = 0 To num_prts -1
    rc = ArsOle.GetServerPrinterInfo (count, name, type)
    ' Process name and type
Next count

.
.
```

GetTypeForDoc

Note: This method intended for use with Visual Basic.

```
short GetTypeForDoc(  
    long          Index,  
    VARIANT *    pType,  
    BSTR *        pExtension )
```

Parameters: **Index**

Specifies the zero-based index of a document within the document list of the active folder. If this value is less than zero, then the open document is used.

pType

Points to a variable to receive the document type of the specified document. The document type will be one of the document type values found in ARSOLEEX.H, such as ARS_OLE_DOC_TYPE_AFP.

pExtension

Points to a BSTR to receive the file extension of the document. This value is returned only if the document type is ARS_OLE_DOC_TYPE_USER_DEF.

Description: Retrieves the document type. If the document type is ARS_OLE_DOC_TYPE_USER_DEF, then the file extension is also retrieved.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: GetDocType

Example

The following example retrieves the document type for the third item in the document list.

```
DIM rc As Integer
DIM type As Variant
DIM ext As String

.
.
.

rc = pArsCtrl->GetTypeForDoc(2, type, ext);
if rc <> ARS_OLE_RC_SUCCESS THEN
    MsgBox "ERROR"
End
Endif

.
.
.
```

IsDocHorzScrollRequired

short **IsDocHorzScrollRequired**(
 VARIANT * **pRequired**)

Parameters: **pRequired**

Points to a variable to receive the result. On return, this variable is set to type VT_I2.

Description: If the width of the document data exceeds the width of the control window, the result variable is set to a non-zero value; otherwise, to zero. The displayed width of the document depends on the inherent width of its data, the type of data (e.g., AFP vs. Linedata), and the zoom factor.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: ScrollDocHorz

Example: The following example initializes the horizontal scroll bar range, shows or hides the scroll bar after a document is opened or the zoom value is changed, and processes WM_HSCROLL messages.

C/C++ Example

```
CArsOle * pArsCtrl;  
CScrollBar * pHorzScrollBar;  
short rc, scroll_code;  
VARIANT scroll_position, required;  
.  
.  
// During initialization:  
  
pHorzScrollBar->SetScrollRange( 0, ARS_OLE_SCROLL_RANGE );  
pHorzScrollBar->ShowScrollBar( FALSE );  
  
// After a document is opened or changing the zoom value:  
  
rc = pArsCtrl->IsDocHorzScrollRequired( &required );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
pHorzScrollBar->ShowScrollBar( required.iVal );  
.  
.  
// While processing a WM_HSCROLL message:  
  
scroll_code = (short)LOWORD(wParam);  
.  
.
```

```

        .
        .
switch ( scroll_code )
{
    case SB_LINELEFT:
        scroll_code = ARS_OLE_SCROLL_LINELEFT;
        break;
    case SB_LINERIGHT:
        scroll_code = ARS_OLE_SCROLL_LINERIGHT;
        break;
    case SB_PAGELEFT:
        scroll_code = ARS_OLE_SCROLL_PAGELEFT;
        break;
    case SB_PAGERIGHT:
        scroll_code = ARS_OLE_SCROLL_PAGERIGHT;
        break;
    case SB_LEFT:
        scroll_code = ARS_OLE_SCROLL_LEFT;
        break;
    case SB_RIGHT:
        scroll_code = ARS_OLE_SCROLL_RIGHT;
        break;
    case SB_THUMBPOSITION:
        scroll_code = ARS_OLE_SCROLL_THUMBPOSITION;
        break;
    case SB_THUMBTRACK:
        scroll_code = ARS_OLE_SCROLL_THUMBTRACK;
        break;
    default:
        scroll_code = ARS_OLE_SCROLL_ENDSCROLL;
}

if ( scroll_code == (short)ARS_OLE_SCROLL_THUMBPOSITION ||
    scroll_code == (short)ARS_OLE_SCROLL_THUMBTRACK )
{
    scroll_position.vt = VT_I2;
    scroll_position.iVal = (short)HIWORD(wParam);
}

rc = pArsCtrl->ScrollDocHorz( scroll_code, &scroll_position );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

pHorzScrollBar->SetScrollPos( (int)scroll_position.iVal );
        .
        .

```

Visual Basic Example

```
Dim rc As Integer
Dim scroll_pos, required As Variant

.
.

' During initialization

sbHorz.Min = 0
sbHorz.Max = ARS_OLE_SCROLL_RANGE
sbHorz.Visible = False

' After a document is opened or changing the zoom value

rc = ArsOle.IsDocHorzScrollRequired (required)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

If required <> 0 Then
    sbHorz.Visible = True
End If

.
.

' During scroll bar Change method

Private Sub sbHorz_Change()
    Dim Diff As Integer
    Dim rc As Integer
    Dim ScrollCode As Integer
    Dim NewPos As Variant
```

```

NewPos = 0
Diff = sbHorz.Value - HorzScrollOld
If Diff = sbHorz.LargeChange Then
    ScrollCode = ARS_OLE_SCROLL_PAGEDOWN
    rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
    HorzScrollOld = NewPos
    sbHorz.Value = NewPos
ElseIf Diff = -sbHorz.LargeChange Then
    ScrollCode = ARS_OLE_SCROLL_PAGEUP
    rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
    HorzScrollOld = NewPos
    sbHorz.Value = NewPos
ElseIf Diff = sbHorz.SmallChange Then
    ScrollCode = ARS_OLE_SCROLL_LINEDOWN
    rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
    HorzScrollOld = NewPos
    sbHorz.Value = NewPos
ElseIf Diff = -sbHorz.SmallChange Then
    ScrollCode = ARS_OLE_SCROLL_LINEUP
    rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
    HorzScrollOld = NewPos
    sbHorz.Value = NewPos
Else
    ScrollCode = ARS_OLE_SCROLL_THUMBPOSITION
    NewPos = sbHorz.Value
    rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
    HorzScrollOld = sbHorz.Value
End If
End Sub

```

Logoff

short **Logoff**()

Description: Logs off from the current server.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: Logon

Example: The following example performs a logoff.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
.  
.  
rc = pArsCtrl->Logoff( );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
  
rc = ArsOle.Logoff ()  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

Logon

```
short Logon(  
    char * pServerName,  
    char * pUserId,  
    char * pPassword )
```

Parameters: **pServerName**

Points to a null-terminated character string containing the name of the server.

pUserId

Points to a null-terminated character string containing the user id.

pPassword

Points to a null-terminated character string containing the password.

Description: If each of **pServerName**, **pUserId**, and **pPassword** are non-NULL and point to a string other than an empty string, a logon is performed with the specified data.

If **pServerName** and **pUserId** point to meaningful strings and **pPassword** points to a single space character, the logon will proceed with no password for the user.

If any of **pServerName**, **pUserId**, and **pPassword** are NULL or point to an empty string, the normal OnDemand Logon dialog box is displayed with the partial information provided. The user can then enter the missing information and complete the logon.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetNumServers, GetServerNames, Logoff, SetLogonReturnOnFailure

Example:

The following example retrieves the names of all servers available for logon, puts them in a ComboBox control, retrieves the chosen server, userid, and password, and performs a logon.

C/C++ Example

```
CArsOle * pArsCtrl;
ArsOleName * pServerNames;
CComboBox * pServersList;
CEdit * pUserId;
CEdit * pPassword;
char server[ sizeof( ArsOleName ) ];
char user[ sizeof( ArsOleName ) ];
char password[ sizeof( ArsOleName ) ];
short rc, j, num_servers;
int index;
VARIANT vari;
.
.
// During dialog initialization:

rc = pArsCtrl->GetNumServers( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_servers = vari.iVal;

pServerNames = new ArsOleName[ max( num_servers, 1 ) ];
rc = pArsCtrl->GetServerNames( (IUnknown*)pServerNames, num_servers );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < num_servers; j++ )
    index = pServersList->AddString( pServerNames[j] );
pServersList->SetCurSel( 0 );
.
.
// During OK button processing:

pServersList->GetWindowText( server, sizeof(server) );
pUserId->GetWindowText( user, sizeof(user) );
pPassword->GetWindowText( password, sizeof(password) );

rc = pArsCtrl->Logon( server, user, password );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.
```

Visual Basic Example

```
Dim rc, count As Integer
Dim num_servers As Variant
Dim Temp As String

.
.

rc = ArsOle.GetNumServers (num_servers)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For count = 0 To num_servers -1
    rc = ArsOle.GetServerName(count, Temp)
    lbServers.AddItem Temp
Next count

.
.

' During OK button processing

rc = ArsOle.Logon (lbServers.List(lbServers.ListItem), txtUser.Value, txtPasswd.Value)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If
```

OnSysColorChange

Note: This method intended for use with C/C++.

void **OnSysColorChange**()

Description: Colors used for OnDemand dialog boxes are updated to the current system colors. The colors used for the document data are not affected.

This method should be called whenever the main window for the OLE container application receives a WM_SYSCOLORCHANGE message.

Return Value: None.

Example: The following example notifies an OnDemand OLE Control when a system color change has occurred.

C/C ++ Example

```
CArsOle * pArsCtrl;  
.  
.  
// During WM_SYSCOLORCHANGE message handling:  
  
pArsCtrl->OnSysColorChange( );  
.  
.
```

OpenDoc

short **OpenDoc**(
 long **Index**,
 char * **pPath**,
 long **ControllId**)

Parameters: **Index**

Specifies the zero-based index of a document within the document list of the active folder.

pPath

Points to a null-terminated character string containing the fully-qualified path of a file containing the document data. If this parameter is NULL, the document data is retrieved from the OnDemand database; if not NULL, the data will be taken from the specified file, but a resource group will be retrieved from the database if required.

ControllId

Specifies the control id of an OnDemand OLE Control. If the value is zero, the control id of this control is used. Refer to "Viewing Multiple Documents for a Single Folder" on page 3 for a discussion of control ids.

Description: The document associated with the specified index in the document list of the active folder of the specified OnDemand OLE Control is opened and displayed in the window of this control.

The reference to a different OnDemand OLE Control allows several windows to simultaneously display documents from a single document list. This avoids the overhead of multiple logon, open folder, and search folder operations. If only one OnDemand OLE Control is being used within an application, the **ControllId** should always be set to zero.

The document retrieval may be cancelled by using the CancelOperation method. If a cancel facility is made available to the user, it may be desirable to call the OpenDoc method on a background thread and allow the user interface thread to monitor the cancellation signal.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetNumFolderDisplayFields, GetFolderDisplayFieldNames, GetNumDocsInList, GetDocDisplayValues, CloseDoc, CancelOperation, WasOperationCancelled, ShowWaitCursorDuringCancelableOperation

Example:

The following example creates a listbox of the folder document list names and associated values and opens the document selected by a user.

C/C++ Example

```
CArsOle * pArsCtrl;  
ArsOleName * pNames;  
ArsOleValue * pValues;  
CListBox * pDocList;  
char * pLine;  
short rc, k, opr, num_fields;  
long j, num_docs;  
int size;  
VARIANT vari;  
.  
.  
// During dialog initialization:  
  
rc = pArsCtrl->GetNumFolderDisplayFields( &vari );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
num_fields = var.iVal;  
  
pNames = new ArsOleName[ max( num_fields, 1 ) ];  
rc = pArsCtrl->GetFolderDisplayFieldNames( (IUnknown*)pNames, num_fields );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
rc = pArsCtrl->GetNumDocsInList( &vari );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
num_docs = var.lVal;  
.  
.
```

```

.
.
pValues = new ArsOleValue[ max( num_fields, 1 ) ];

size = num_fields * ( sizeof(ArsOleName) + sizeof(ArsOleValue) + 5 );
pLine = new char[ size ];
for ( j = 0, pLine[0] = '\0'; j < num_docs; j++ )
{
    rc = pArsCtrl->GetDocDisplayValues( j, pValues, num_fields );
    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;

    for ( k = 0; k < num_fields; k++ )
    {
        strcat( pLine, pNames[k] );
        strcat( pLine, " = " );
        strcat( pLine, pValues[k] );
        if ( k < num_fields - 1 )
            strcat( pLine, ", " );
    }
    pDocList->InsertString( -1, pLine );
}
pDocList->SetCurSel( 0 );
.
.
// During OK button processing:

rc = pArsCtrl->OpenDoc( (long)pDocList->GetCurSel( ) , NULL, 0 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```
Dim rc, count, i, j As Integer
Dim num_fields, num_docs As Variant
Dim Names() As String
Dim Line As String
Dim Temp As String

.
.

rc = ArsOle.GetNumFolderDisplayFields(num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

ReDim Names(num_fields -1)

For count = 0 To num_fields -1
    rc = ArsOle.GetFolderDisplayFieldName(count, Temp)
    Names(count) = Temp
Next count

rc = ArsOle.GetNumDocsInList(num_docs)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For j = 0 To num_docs -1
    For i = 0 To num_fields -1
        rc = ArsOle.GetDocDisplayValue(j, i, Temp)

        Line = Line + Names(i) + " = " + Temp
        If i < num_fields Then
            Line = Line + ", "
        End If
    Next i

    lbDocs.AddItem Line
Next j

.
.

'During OK button processing:

rc = ArsOle.OpenDoc (lbDocs.List(lbDocs.ListIndex), "", 0)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If
```

OpenFolder

short **OpenFolder**(
 char * **pFolderName**)

Parameters: **pFolderName**

Points to a null-terminated character string containing the name of the folder.

Description: If **pFolderName** is non-NULL and points to a string other than an empty string, the named folder is opened.

 If **pFolderName** is NULL or points to an empty string, the normal OnDemand Open Folder dialog box is displayed. The user can then select the folder and complete the open.

The opened folder becomes the active folder. The OnDemand Folder dialog box is initially hidden. It can be displayed by using the ShowFolder method.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: GetNumFolders, GetFolderNames, CloseFolder, CloseAllFolders

Example:

The following example retrieves the names of all folders available for the current server, puts them in a ComboBox control, retrieves the chosen folder, and performs an open for that folder.

C/C++ Example

```
CArsOle * pArsCtrl;
ArsOleName * pFolderNames;
CComboBox * pFoldersList;
char folder[ sizeof( ArsOleName ) ];
short rc, j, num_folders;
int index;
VARIANT vari;
.
.
// During dialog initialization:

rc = pArsCtrl->GetNumFolders( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_folders = var.iVal;

pFolderNames = new ArsOleName[ max( num_folders, 1 ) ];
rc = pArsCtrl->GetFolderNames( (IUnknown*)pFolderNames, num_folders );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < num_folders; j++ )
    index = pFoldersList->AddString( pFolderNames[j] );
pFoldersList->SetCurSel( 0 );
.
.
// During OK button processing:

pFoldersList->GetWindowText( folder, sizeof(folder) );

rc = pArsCtrl->OpenFolder( folder );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.
```

Visual Basic Example

```
Dim rc, count As Integer
Dim num_folders As Variant
Dim Temp As String

.
.

rc = ArsOle.GetNumFolders (num_folders)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For count = 0 To num_folders -1
    rc = ArsOle.GetFolderName(count, Temp)
    lbFolders.AddItem Temp
Next count

.
.

' During OK button processing

rc = ArsOle.OpenFolder (lbFolders.List(lbFolders.ListItem))
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If
```

PrintDoc

```
short PrintDoc(  
    long      Index,  
    long      Page,  
    char *    pPrinterName,  
    boolean   LocalPrinter,  
    short     Copies,  
    short     Orientation,  
    float     TopMargin,  
    float     BottomMargin,  
    float     LeftMargin,  
    float     RightMargin,  
    boolean   MarginsInMillimeters )
```

Parameters: **Index**

Specifies the zero-based index of a document within the document list of the active folder. If this value is less than zero, the open document is printed. If a local printer is specified, only the open document may be printed.

Page

Specifies the page number to be printed. If this parameter is less than or equal to zero, the entire document is printed. If a server printer is specified, this parameter is ignored and the entire document is printed.

pPrinterName

Points to a null-terminated character string containing the name of a local or server printer. For local printers, the name should be the same as that typically displayed in a printer selection list (e.g., IBM Laser Printer 4019 on LPT1:). For server printers, the name should be one of those defined for the current server in the OnDemand database.

LocalPrinter

If non-zero, indicates that the name specified for **pPrinterName** is a local printer; if zero, that it specifies a server printer. If local, only the open document may be printed. If server, the orientation and margin values are ignored and the entire document is printed.

Copies

Specifies the number of copies to be printed. This value must be between 1 and 100.

Orientation

Specifies the page orientation. This must be one of the following orientation values found in ARSOLEEX.H:

```
ARS_OLE_ORIENTATION_PORTRAIT
ARS_OLE_ORIENTATION_LANDSCAPE
ARS_OLE_ORIENTATION_BEST_FIT
ARS_OLE_ORIENTATION_ASIS
```

This parameter is ignored if a server printer is specified.

TopMargin

Specifies the amount of space for the top page margin. This parameter is ignored if a server printer is specified.

BottomMargin

Specifies the amount of space for the bottom page margin. This parameter is ignored if a server printer is specified.

LeftMargin

Specifies the amount of space for the left page margin. This parameter is ignored if a server printer is specified.

RightMargin

Specifies the amount of space for the right page margin. This parameter is ignored if a server printer is specified.

MarginsInMillimeters

If non-zero, indicates that the margin values are specified in millimeters; if zero, that they are specified in inches. This parameter is ignored if a server printer is specified.

Description: The page(s) of the specified document are printed.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetNumDocsInList, OpenDoc

Example: The following example prints page 5 of the open document on a local printer.

C/C++ Example

```

CArsOle * pArsCtrl;
short rc;
.
.
rc = pArsCtrl->PrintDoc( -1,
                        5,
                        "Acrobat PDFWriter on DISK:",
                        TRUE,
                        1,
                        ARS_OLE_ORIENTATION_BEST_FIT,
                        0.5,
                        0.5,
                        0.5,
                        0.5,
                        FALSE );

if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```
Dim rc As Integer
```

```
.  
.
```

```
rc = ArsOle.PrintDoc (-1, _  
                    5, _  
                    "Acrobat PDFWriter on DISK:", _  
                    True, _  
                    1, _  
                    ARS_OLE_ORIENTATION_BEST_FIT, _  
                    0.5, _  
                    0.5, _  
                    0.5, _  
                    0.5, _  
                    False)
```

```
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If
```

```
.  
.
```

RetrieveDoc

```
short RetrieveDoc(  
    long    Index,  
    char *  pDocPath,  
    char *  pResGrpPath,  
    char *  pCombinedPath )
```

Parameters: **Index**

Specifies the zero-based index of a document within the document list of the active folder.

pDocPath

Points to a null-terminated character string containing the fully-qualified path of a file to contain the document data. If this parameter is NULL, no data is written.

pResGrpPath

Points to a null-terminated character string containing the fully-qualified path of a file to contain the resource group data. If this parameter is NULL, no data is written.

pCombinedPath

Points to a null-terminated character string containing the fully-qualified path of a file to contain the combined resource group and document data. If this parameter is NULL, no data is written.

Description: The data for the indicated document and its associated resource group, if any, are retrieved from the OnDemand database and written to the indicated files. If any of the files already exist, the data is appended to the files. In the combined file, the data for the document is placed after the data for the resource group. Any combination of **pDocPath**, **pResGrpPath**, and **pCombinedPath** may be specified.

The document retrieval may be cancelled by using the CancelOperation method. If a cancel facility is made available to the user, it may be desirable to call the RetrieveDoc method on a background thread and allow the user interface thread to monitor the cancellation signal.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetNumDocsInList

Example: The following example retrieves the data for all files in a document list, copying all document data to one file and the resource group data to a different file.

C/C++ Example

```

CArsOle * pArsCtrl;
long num_docs, doc_num;
short rc;

.
.
rc = pArsCtrl->GetNumDocsInList( &num_docs );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( doc_num = 0; doc_num < num_docs; doc_num++ )
{
    rc = pArsCtrl->RetrieveDoc( doc_num,
                              "C:\\FILES\\DATA.DOC",
                              doc_num == 0 ? "C:\\FILES\\DATA.RG" : NULL,
                              NULL );
    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;
}

.
.

```

Visual Basic Example

```

Dim rc, count As Integer
Dim num_docs As Variant

.
.

rc = ArsOle.GetNumDocsInList (num_docs)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

For count = 0 To num_docs - 1
    If count = 0 Then
        rc = ArsOle.RetrieveDoc (count, _
                                "C:\FILES\DATA.DOC", _
                                "C:\FILES\DATA.RG", _
                                "")
        If rc <> ARS_OLE_RC_SUCCESS Then
            MsgBox "ERROR"
        End
        End If
    Else
        rc = ArsOle.RetrieveDoc (count, _
                                "C:\FILES\DATA.DOC", _
                                "", _
                                "")
        If rc <> ARS_OLE_RC_SUCCESS Then
            MsgBox "ERROR"
        End
        End If
    End If
Next count

.
.

```

ScrollDocHorz

short **ScrollDocHorz**(
 short **Type**,
 VARIANT * **pPosition**)

Parameters:

Type

Specifies a scrollbar code that identifies the type of scrolling required. This must be one of the following scroll types found in ARSOLEEX.H:

```
ARS_OLE_SCROLL_LINELEFT  
ARS_OLE_SCROLL_LINERIGHT  
ARS_OLE_SCROLL_PAGELEFT  
ARS_OLE_SCROLL_PAGERIGHT  
ARS_OLE_SCROLL_LEFT  
ARS_OLE_SCROLL_RIGHT  
ARS_OLE_SCROLL_THUMBPOSITION  
ARS_OLE_SCROLL_THUMBTRACK  
ARS_OLE_SCROLL_ENDSCROLL
```

pPosition

Points to a variable that contains and/or will contain the scroll position. When **Type** is ARS_OLE_SCROLL_THUMBPOSITION or ARS_OLE_SCROLL_THUMBTRACK, the variable must contain the position to which to scroll. For all types, on return this variable contains the current scroll position. This variable should be set to type VT_I2.

Description: The document data is horizontally scrolled as indicated by **Type** and the current scroll position is returned in the variable pointed to by **pPosition**. On input and on return, the position value assumes that the horizontal scroll range has been set to ARS_OLE_SCROLL_RANGE. If a different value is used, the units should be converted before and after the call.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: IsDocHorzScrollRequired, ScrollDocVert

Example: The following example initializes the horizontal scroll bar range, shows or hides the scroll bar after a document is opened or the zoom value is changed, and processes WM_HSCROLL messages.

C/C++ Example

```
CArsOle * pArsCtrl;  
CScrollBar * pHorzScrollBar;  
short rc, scroll_code;  
VARIANT scroll_position, required;  
.  
.  
// During initialization:  
  
pHorzScrollBar->SetScrollRange( 0, ARS_OLE_SCROLL_RANGE );  
pHorzScrollBar->ShowScrollBar( FALSE );  
.  
.
```



```

// After a document is opened or changing the zoom value:

rc = pArsCtrl->IsDocHorzScrollRequired( &required );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

pHorzScrollBar->ShowScrollBar( required.iVal );
.
.
// While processing a WM_HSCROLL message:

scroll_code = (short)LOWORD(wParam);

switch ( scroll_code )
{
    case SB_LINELEFT:
        scroll_code = ARS_OLE_SCROLL_LINELEFT;
        break;
    case SB_LINERIGHT:
        scroll_code = ARS_OLE_SCROLL_LINERIGHT;
        break;
    case SB_PAGELEFT:
        scroll_code = ARS_OLE_SCROLL_PAGELEFT;
        break;
    case SB_PAGERIGHT:
        scroll_code = ARS_OLE_SCROLL_PAGERIGHT;
        break;
    case SB_LEFT:
        scroll_code = ARS_OLE_SCROLL_LEFT;
        break;
    case SB_RIGHT:
        scroll_code = ARS_OLE_SCROLL_RIGHT;
        break;
    case SB_THUMBPOSITION:
        scroll_code = ARS_OLE_SCROLL_THUMBPOSITION;
        break;
    case SB_THUMBTRACK:
        scroll_code = ARS_OLE_SCROLL_THUMBTRACK;
        break;
    default:
        scroll_code = ARS_OLE_SCROLL_ENDSCROLL;
}

if ( scroll_code == (short)ARS_OLE_SCROLL_THUMBPOSITION ||
    scroll_code == (short)ARS_OLE_SCROLL_THUMBTRACK )
{
    scroll_position.vt = VT_I2;
    scroll_position.iVal = (short)HIWORD(wParam);
}

```

```
rc = pArsCtrl->ScrollDocHorz( scroll_code, &scroll_position );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

pHorzScrollBar->SetScrollPos( (int)scroll_position.iVal );
.
.
```

Visual Basic Example

```
Dim rc As Integer
Dim scroll_pos, required As Variant
.
.
' During initialization

sbHorz.Min = 0
sbHorz.Max = ARS_OLE_SCROLL_RANGE
sbHorz.Visible = False

' After a document is opened or changing the zoom value

rc = ArsOle.IsDocHorzScrollRequired (required)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

If required <> 0 Then
    sbHorz.Visible = True
End If

.
.

' During scroll bar Change method

Private Sub sbHorz_Change()
    Dim Diff As Integer
    Dim rc As Integer
    Dim ScrollCode As Integer
    Dim NewPos As Variant
```

```

NewPos = 0
Diff = sbHorz.Value - HorzScrollOld
If Diff = sbHorz.LargeChange Then
    ScrollCode = ARS_OLE_SCROLL_PAGEDOWN
    rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
    HorzScrollOld = NewPos
    sbHorz.Value = NewPos
ElseIf Diff = -sbHorz.LargeChange Then
    ScrollCode = ARS_OLE_SCROLL_PAGEUP
    rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
    HorzScrollOld = NewPos
    sbHorz.Value = NewPos
ElseIf Diff = sbHorz.SmallChange Then
    ScrollCode = ARS_OLE_SCROLL_LINEDOWN
    rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
    HorzScrollOld = NewPos
    sbHorz.Value = NewPos
ElseIf Diff = -sbHorz.SmallChange Then
    ScrollCode = ARS_OLE_SCROLL_LINEUP
    rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
    HorzScrollOld = NewPos
    sbHorz.Value = NewPos
Else
    ScrollCode = ARS_OLE_SCROLL_THUMBPOSITION
    NewPos = sbHorz.Value
    rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
    HorzScrollOld = sbHorz.Value
End If
End Sub

```

ScrollDocVert

short **ScrollDocVert**(
 short **Type**,
 VARIANT * **pPosition**)

Parameters: **Type**

Specifies a scrollbar code that identifies the type of scrolling required. This must be one of the following scroll types found in ARSOLEEX.H:

ARS_OLE_SCROLL_LINEUP
ARS_OLE_SCROLL_LINEDOWN
ARS_OLE_SCROLL_PAGEUP
ARS_OLE_SCROLL_PAGEDOWN
ARS_OLE_SCROLL_TOP
ARS_OLE_SCROLL_BOTTOM
ARS_OLE_SCROLL_THUMBPOSITION
ARS_OLE_SCROLL_THUMBTRACK
ARS_OLE_SCROLL_ENDSCROLL

pPosition

Points to a variable that contains and/or will contain the scroll position. When **Type** is ARS_OLE_SCROLL_THUMBPOSITION or ARS_OLE_SCROLL_THUMBTRACK, the variable must contain the position to which to scroll. For all types, on return this variable contains the current scroll position. This variable should be set to type VT_I2.

Description: The document data is vertically scrolled as indicated by **Type** and the current scroll position is returned in the variable pointed to by **pPosition**. On input and on return, the position value assumes that the vertical scroll range has been set to ARS_OLE_SCROLL_RANGE. If a different value is used, the units should be converted before and after the call.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: ScrollDocHorz

Example:

The following example initializes the vertical scroll bar range and processes WM_VSCROLL messages.

C/C++ Example

```
CArsOle * pArsCtrl;
CScrollBar * pVertScrollBar;
short rc, scroll_code;
VARIANT scroll_position;
.
.
// During initialization:

pVertScrollBar->SetScrollRange( 0, ARS_OLE_SCROLL_RANGE );
pVertScrollBar->ShowScrollBar( TRUE );

// While processing a WM_VSCROLL message:

scroll_code = (short)LOWORD(wParam);

switch ( scroll_code )
{
    case SB_LINEUP:
        scroll_code = ARS_OLE_SCROLL_LINEUP;
        break;
    case SB_LINEDOWN:
        scroll_code = ARS_OLE_SCROLL_LINEDOWN;
        break;
    case SB_PAGEUP:
        scroll_code = ARS_OLE_SCROLL_PAGEUP;
        break;
    case SB_PAGEDOWN:
        scroll_code = ARS_OLE_SCROLL_PAGEDOWN;
        break;
    case SB_TOP:
        scroll_code = ARS_OLE_SCROLL_TOP;
        break;
    case SB_BOTTOM:
        scroll_code = ARS_OLE_SCROLL_BOTTOM;
        break;
    case SB_THUMBPOSITION:
        scroll_code = ARS_OLE_SCROLL_THUMBPOSITION;
        break;
    case SB_THUMBTRACK:
        scroll_code = ARS_OLE_SCROLL_THUMBTRACK;
        break;
    default:
        scroll_code = ARS_OLE_SCROLL_ENDSCROLL;
}
.
.
```

```

        .
        .
    if ( scroll_code == (short)ARS_OLE_SCROLL_THUMBPOSITION ||
        scroll_code == (short)ARS_OLE_SCROLL_THUMBTRACK )
    {
        scroll_position.vt = VT_I2;
        scroll_position.iVal = (short)HIWORD(wParam);
    }

    rc = pArsCtrl->ScrollDocVert( scroll_code, &scroll_position );
    if ( rc != ARS_OLE_RC_SUCCESS )
        ERROR;

    pVertScrollBar->SetScrollPos( (int)scroll_position.iVal );
    .
    .

```

Visual Basic Example

```

Dim rc As Integer
Dim scroll_pos, required As Variant

```

```

        .
        .

' During initialization

sbVert.Min = 0
sbVert.Max = ARS_OLE_SCROLL_RANGE
sbVert.Visible = True

        .
        .

' During scroll bar Change method

Private Sub sbVert_Change()
    Dim Diff As Integer
    Dim rc As Integer
    Dim ScrollCode As Integer
    Dim NewPos As Variant

```

```

NewPos = 0
Diff = sbVert.Value - VertScrollOld
If Diff = sbVert.LargeChange Then
    ScrollCode = ARS_OLE_SCROLL_PAGEDOWN
    rc = ArsOle.ScrollDocVert(ScrollCode, NewPos)
    VertScrollOld = NewPos
    sbVert.Value = NewPos
ElseIf Diff = -sbVert.LargeChange Then
    ScrollCode = ARS_OLE_SCROLL_PAGEUP
    rc = ArsOle.ScrollDocVert(ScrollCode, NewPos)
    VertScrollOld = NewPos
    sbVert.Value = NewPos
ElseIf Diff = sbVert.SmallChange Then
    ScrollCode = ARS_OLE_SCROLL_LINEDOWN
    rc = ArsOle.ScrollDocVert(ScrollCode, NewPos)
    VertScrollOld = NewPos
    sbVert.Value = NewPos
ElseIf Diff = -sbVert.SmallChange Then
    ScrollCode = ARS_OLE_SCROLL_LINEUP
    rc = ArsOle.ScrollDocVert(ScrollCode, NewPos)
    VertScrollOld = NewPos
    sbVert.Value = NewPos
Else
    ScrollCode = ARS_OLE_SCROLL_THUMBPOSITION
    NewPos = sbVert.Value
    rc = ArsOle.ScrollDocVert(ScrollCode, NewPos)
    VertScrollOld = sbVert.Value
End If
End Sub

```

SearchFolder

short **SearchFolder**(
 boolean **Append**)

Parameters: **Append**

If non-zero, indicates that the results of the search are to be appended to the existing document list; otherwise, that the results of the search are to replace the existing document list.

Description: The active folder is searched with the current field values. These values were set by default, by the user, or by the SetFolderSearchFieldData method.

The search operation may be cancelled by using the CancelOperation method. If a cancel facility is made available to the user, it may be desirable to call the SearchFolder method on a background thread and allow the user interface thread to monitor the cancellation signal.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: OpenFolder, GetNumFolderSearchFields, GetFolderSearchFieldNames, SetFolderSearchFieldData, CancelOperation, WasOperationCancelled, ShowWaitCursorDuringCancelableOperation

Example: The following example retrieves the names of the active folder search fields, gives a user the opportunity to set the values for these fields, and initiates a search of the folder.

C/C++ Example

```

CArsOle * pArsCtrl;
ArsOleName * pNames;
CListBox * pFieldList, * pOprList;
CEdit * pValue1, * pValue2;
char name[ sizeof( ArsOleName ) ];
char value1[ sizeof( ArsOleValue ) ];
char value2[ sizeof( ArsOleValue ) ];
short rc, j, opr, num_fields;
VARIANT vari;
.
.
struct _OprMap
{
    short code;
    char * pText;
} OprMap

static OprMap Oprs[] =
{ { ARS_OLE_OPR_EQUAL, "Equal" },
  { ARS_OLE_OPR_NOT_EQUAL, "Not Equal" },
  .
  .
  { ARS_OLE_OPR_LIKE, "Like" },
  { ARS_OLE_OPR_NOT_LIKE, "Not Like" } };

#define NUM_OPRS ( sizeof(Oprs) / sizeof(OprMap) )
.
.
// During dialog initialization:

rc = pArsCtrl->GetNumFolderSearchFields( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_fields = var.iVal;

pNames = new ArsOleName[ max( num_fields, 1 ) ];
rc = pArsCtrl->GetFolderSearchFieldNames( (IUnknown*)pNames, num_fields );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < num_fields; j++ )
    pFieldList->InsertString( -1, pNames[j] );
pFieldList->SetCurSel( 0 );

for ( j = 0; j < NUM_OPRS; j++ )
{
    pOprList->InsertString( -1, Oprs[j].pText );
    pOprList->SetItemData( j, (DWORD)Oprs[j].code );
}
pOprList->SetCurSel( 0 );
.
.

```

```

// During SET FIELD button processing:

pFieldList->GetText( pFieldList->GetCurSel( ), name );
opr = (short)pOprList->GetItemData( pOprList->GetCurSel( ) );
pValue1->GetWindowText( value1, sizeof(value1) );
pValue2->GetWindowText( value2, sizeof(value2) );

rc = pArsCtrl->SetFolderSearchFieldData( name, opr, value1, value2 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

// During OK button processing:

rc = pArsCtrl->SearchFolder( FALSE );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```
Dim rc, count, i, j As Integer
Dim num_fields, num_docs As Variant
Dim Names() As String
Dim Line As String
Dim Temp As String
Dim Oprs As Variant

.
.

Oprs = Array ("Equal", "Not Equal", ..., "Like", "Not Like")

rc = ArsOle.GetNumFolderSearchFields(num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

ReDim Names(num_fields -1)

For count = 0 To num_fields -1
    rc = ArsOle.GetFolderSearchFieldName(count, Temp)
    Names(count) = Temp
Next count

for count = 0 To num_fields -1
    lbFieldList.AddItem Names(count)
Next count

for count = 0 To UBound(Oprs) -1
    lbOprList.AddItem (Oprs(count))
Next count
```

```

' During SET FIELD button processing
rc = ArsOle.SetFolderSearchFieldData (lbFieldList.List(lbFieldList.ListIndex),
                                     lbOprList.ListIndex,
                                     txtValue1.Value,
                                     txtValue2.Value)

If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If
.
.
.
'During OK button processing:

rc = ArsOle.SearchFolder (False)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

SetDefaultFolderSearchFields

short **SetDefaultFolderSearchFields**()

Description: The search fields for the active folder are set to their default values.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: OpenFolder, SearchFolder

Example: The following example sets the search fields for the active folder to their default values.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
.  
.  
rc = pArsCtrl->SetDefaultFolderSearchFields( );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
  
rc = ArsOle.SetDefaultFolderSearchFields ( )  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

SetDocBackgroundColor

short **SetDocBackgroundColor**(
 short **Color**)

Parameters: **Color**

Specifies the new document background color. This must be one of the following color values found in ARSOLEEX.H:

```
ARS_OLE_COLOR_WHITE  
ARS_OLE_COLOR_BLACK  
ARS_OLE_COLOR_RED  
ARS_OLE_COLOR_BLUE  
ARS_OLE_COLOR_GREEN  
ARS_OLE_COLOR_YELLOW  
ARS_OLE_COLOR_GREY
```

Description: The document is displayed with the new background color.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: SetDocBackgroundColor

Example: The following example sets the document background color to grey.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
  
.  
.  
rc = pArsCtrl->SetDocBackgroundColor( ARS_OLE_COLOR_GREY );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
  
rc = ArsOle.SetDocBackgroundColor (ARS_OLE_COLOR_GREY)  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

SetDocCurrentPage

short **SetDocCurrentPage**(
 long **Page**)

Parameters: **Page**

Specifies the page number to be made the current page number of the open document.

Description: The current page number of the open document is set to the specified page and the control window is repainted with the data for that page.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetDocCurrentPage, GetDocNumPages

Example: The following example sets the current page number of the open document and updates the current scroll positions.

C/C++ Example

```
CArsOle * pArsCtrl;  
CScrollBar * pHorzScrollBar, * pVertScrollBar;  
short rc;  
VARIANT horz_position, vert_position;  
.  
.  
rc = pArsCtrl->SetDocCurrentPage( 46 );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
.  
.  
  
rc = pArsCtrl->GetDocScrollPositions( &horz_position, &vert_position );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
pHorzScrollBar->SetScrollPos( (int)horz_position.iVal );  
pVertScrollBar->SetScrollPos( (int)vert_position.iVal );  
.  
.
```


Visual Basic Example

```
Dim rc As Integer
Dim horz_pos, vert_pos As Variant

.
.

rc = ArsOle.SetDocCurrentPage( 46 )
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

rc = ArsOle.GetDocScrollPositions( horz_pos, vert_pos )
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

sbHorz.Value = horz_pos
sbVert.Value = vert_pos

.
.
```

SetDocImageColor

short **SetDocImageColor**(
 short **Color**)

Parameters: **Color**

Specifies the new document image color. This must be one of the following color values found in ARSOLEEX.H:

ARS_OLE_COLOR_BLACK
ARS_OLE_COLOR_RED
ARS_OLE_COLOR_BLUE
ARS_OLE_COLOR_GREEN
ARS_OLE_COLOR_YELLOW
ARS_OLE_COLOR_GREY
ARS_OLE_COLOR_MAGENTA
ARS_OLE_COLOR_CYAN

Description: The document is displayed with the new image color.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetDocImageColor

Example: The following example sets the document image color to red.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
  
.  
.  
  
rc = pArsCtrl->SetDocImageColor( ARS_OLE_COLOR_RED );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
  
rc = ArsOle.SetDocImageColor (ARS_OLE_COLOR_RED)  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

SetDocImageIntensity

short **SetDocImageIntensity**(
 short **Intensity**)

Parameters: **Intensity**

Specifies the new document image intensity. This must be one of the following intensity values found in ARSOLEEX.H:

 ARS_OLE_INTENSITY_NORMAL
 ARS_OLE_INTENSITY_LIGHT
 ARS_OLE_INTENSITY_NONE

Description: The document is displayed with the new image intensity.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: GetDocImageIntensity

Example: The following example sets the document image intensity to light.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
.  
.  
rc = pArsCtrl->SetDocImageIntensity( ARS_OLE_INTENSITY_LIGHT );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
  
rc = ArsOle.SetDocImageIntensity (ARS_OLE_INTENSITY_LIGHT)  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

SetDocRotation

short **SetDocRotation**(
 short **Rotation**)

Parameters: **Rotation**

Specifies the new document rotation. This must be one of the following rotation values found in ARSOLEEX.H:

ARS_OLE_ROTATION_0
ARS_OLE_ROTATION_90
ARS_OLE_ROTATION_180
ARS_OLE_ROTATION_270

Description: The document is displayed at the new rotation.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetDocRotation

Example: The following example rotates the document to 90 degrees.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
.  
.  
rc = pArsCtrl->SetDocRotation( ARS_OLE_ROTATION_90 );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
.  
.  
  
rc = ArsOle.SetDocRotation (ARS_OLE_ROTATION_90)  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

SetDocZoom

```
short SetDocZoom(  
    short      ZoomPercent,  
    VARIANT *  pHorzPosition,  
    VARIANT *  pVertPosition )
```

Parameters: **ZoomPercent**

Specifies the new zoom percent to be set.

pHorzPosition

Points to a variable to receive the new horizontal scroll position. On return, this variable is set to type VT_I2.

pVertPosition

Points to a variable to receive the new vertical scroll position. On return, this variable is set to type VT_I2.

Description: The document is displayed at the new zoom percent and the new scroll positions are returned in the specified variables. The scroll positions assume that the scroll ranges have been set to ARS_OLE_SCROLL_RANGE.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetDocZoom

Example: The following example sets a new zoom value and repositions the scroll bars.

C/C++ Example

```
CArsOle * pArsCtrl;  
CScrollBar * pHorzScrollBar, * pVertScrollBar;  
short rc;  
VARIANT horz_position, vert_position;  
.  
.  
// During initialization:  
  
pHorzScrollBar->SetScrollRange( 0, ARS_OLE_SCROLL_RANGE );  
pVertScrollBar->SetScrollRange( 0, ARS_OLE_SCROLL_RANGE );  
.  
.  
// When required to double the zoom factor:  
  
rc = pArsCtrl->SetDocZoom( 200, &horz_position, &vert_position );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
pHorzScrollBar->SetScrollPos( (int)horz_position.iVal );  
pVertScrollBar->SetScrollPos( (int)vert_position.iVal );  
.  
.
```

Visual Basic Example

```
Dim rc As Integer
Dim horz_pos, vert_pos As Variant

.
.

' During initialization

sbHorz.Min = 0
sbHorz.Max = ARS_OLE_SCROLL_RANGE
sbVert.Min = 0
sbVert.Max = ARS_OLE_SCROLL_RANGE

.
.

' When required to double the zoom factor

rc = ArsOle.SetDocZoom (200, horz_pos, vert_pos)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

sbHorz.Value = horz_pos
sbVert.Value = vert_pos
```

SetFolderCloseMemoryRelease

short **SetFolderCloseMemoryRelease**(
 boolean **Release**)

Parameters: **Release**

If non-zero, indicates that all memory associated with a folder is to be released when the folder is closed; otherwise, that the memory is to be retained.

Description: Determines whether memory is to be released when a folder is closed. By default, the setting is false.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: CloseFolder

Example: The following example causes memory to be released when a folder is closed.

C/C ++ Example

```
CArs01e * pArsCtrl;  
.  
.  
.  
pArsCtrl->SetFolderCloseMemoryRelease( TRUE );  
.  
.  
.
```

Visual Basic Example

```
.  
.  
.  
Ars01e.SetFolderCloseMemoryRelease (True)  
.  
.  
.
```

SetFolderSearchFieldData

```
short SetFolderSearchFieldData(  
    char *    pFieldName,  
    short     Operator,  
    char *    pValue1,  
    char *    pValue2 )
```

Parameters: **pFieldName**

Points to a null-terminated character string containing the name of a search field for the active folder.

Operator

Specifies the search operator to be used. This must be one of the following operator values found in ARSOLEEX.H:

```
ARS_OLE_OPR_EQUAL  
ARS_OLE_OPR_NOT_EQUAL  
ARS_OLE_OPR_LESS_THAN  
ARS_OLE_OPR_LESS_THAN_OR_EQUAL  
ARS_OLE_OPR_GREATER_THAN  
ARS_OLE_OPR_GREATER_THAN_OR_EQUAL  
ARS_OLE_OPR_BETWEEN  
ARS_OLE_OPR_NOT_BETWEEN  
ARS_OLE_OPR_IN  
ARS_OLE_OPR_NOT_IN  
ARS_OLE_OPR_LIKE  
ARS_OLE_OPR_NOT_LIKE
```

pValue1

Points to a null-terminated character string containing the first, or only, value to be set for the field.

pValue2

Points to a null-terminated character string containing the second value to be set for the field. This parameter is ignored unless the operator is ARS_OLE_OPR_BETWEEN or ARS_OLE_OPR_NOT_BETWEEN.

Description: The search operator and value(s) are set for the specified field for the active folder.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: GetNumFolderSearchFields, GetFolderSearchFieldNames, SearchFolder

Example:

The following example retrieves the names of the active folder search fields, gives a user the opportunity to set the values for these fields, and initiates a search of the folder.

C/C++ Example

```
CArsOle * pArsCtrl;
ArsOleName * pNames;
CListBox * pFieldList, * pOprList;
CEdit * pValue1, * pValue2;
char name[ sizeof( ArsOleName ) ];
char value1[ sizeof( ArsOleValue ) ];
char value2[ sizeof( ArsOleValue ) ];
short rc, j, opr, num_fields;
VARIANT vari;
.
.
struct _OprMap
{
    short code;
    char * pText;
} OprMap

static OprMap Oprs[] =
{ { ARS_OLE_OPR_EQUAL,          "Equal" },
  { ARS_OLE_OPR_NOT_EQUAL,     "Not Equal" },
  .
  .
  { ARS_OLE_OPR_LIKE,          "Like" },
  { ARS_OLE_OPR_NOT_LIKE,     "Not Like" } };

#define NUM_OPRS ( sizeof(Oprs) / sizeof(OprMap) )
.
.
```

```

// During dialog initialization:

rc = pArsCtrl->GetNumFolderSearchFields( &vari );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
num_fields = var.iVal;

pNames = new ArsOleName[ max( num_fields, 1 ) ];
rc = pArsCtrl->GetFolderSearchFieldNames( (IUnknown*)pNames, num_fields );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;

for ( j = 0; j < num_fields; j++ )
    pFieldList->InsertString( -1, pNames[j] );
pFieldList->SetCurSel( 0 );

for ( j = 0; j < NUM_OPRS; j++ )
{
    pOprList->InsertString( -1, Oprs[j].pText );
    pOprList->SetItemData( j, (DWORD)Oprs[j].code );
}
pOprList->SetCurSel( 0 );
.
.

// During SET FIELD button processing:

pFieldList->GetText( pFieldList->GetCurSel( ), name );
opr = (short)pOprList->GetItemData( pOprList->GetCurSel( ) );
pValue1->GetWindowText( value1, sizeof(value1) );
pValue2->GetWindowText( value2, sizeof(value2) );

rc = pArsCtrl->SetFolderSearchFieldData( name, opr, value1, value2 );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

// During OK button processing:

rc = pArsCtrl->SearchFolder( FALSE );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```

Dim rc, count, i, j As Integer
Dim num_fields, num_docs As Variant
Dim Names() As String
Dim Line As String
Dim Temp As String
Dim Oprs As Variant

.
.

Oprs = Array ("Equal", "Not Equal", ..., "Like", "Not Like")

rc = ArsOle.GetNumFolderSearchFields(num_fields)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

ReDim Names(num_fields)

For count = 1 To num_fields
    rc = ArsOle.GetFolderSearchFieldName(count, Temp)
    Names(count) = Temp
Next count

for count = 1 To num_fields
    lbFieldList.AddItem Names(count)
Next count

for count = 1 To UBound(Oprs)
    lbOprList.AddItem (Oprs(count))
Next count

.
.
.

```

```

.
.
.
' During SET FIELD button processing
rc = ArsOle.SetFolderSearchFieldData (lbFieldList.List(lbFieldList.ListIndex),
                                      lbOprList.ListIndex,
                                      txtValue1.Value,
                                      txtValue2.Value)

If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

'During OK button processing:

rc = ArsOle.SearchFolder (False)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

```

SetLogonReturnOnFailure

short **SetLogonReturnOnFailure**(
 boolean **Return**)

Parameters: **Return**

If non-zero, indicates that control is to be returned when a logon fails; otherwise, that the OnDemand Logon dialog box is to be displayed when a logon fails.

Description: Determines the action to be taken when a logon fails. By default, the setting is false.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: Logon

Example: The following example sets the action to be taken when a logon fails.

C/C ++ Example

```
CArs01e * pArsCtrl;  
.  
.  
.  
pArsCtrl->SetLogonReturnOnFailure( TRUE );  
.  
.  
.
```

Visual Basic Example

```
.  
.  
.  
Ars01e.SetLogonReturnOnFailure (True)  
.  
.  
.
```

SetRightButtonMenu

short **SetRightButtonMenu**(
 char * **pMenuData**,
 VARIANT * **pErrorPosition**)

Parameters: **pMenuData**

Specifies the data to be used to create a menu. If this parameter is null, any existing menu is deleted.

pErrorPosition

Points to a variable to receive an error position if the menu cannot be created. In that case, the variable is set to type VT_I4.

Description:

OnDemand creates a menu to be displayed when the user clicks the right mouse button in the document window. This menu replaces any existing menu. The format of the menu is determined by the data specified by the **pMenuData** parameter. Complex menus containing commands, separators, and submenus can be created.

The menu data consists of a set of menu items separated by a newline character (x'0A'). There can be a maximum of ARS_OLE_MAX_MENU_ITEMS items, each of which can contain no more than ARS_OLE_MAX_MENU_ITEM_LEN characters. The items must appear in the order in which they are to be displayed in the menu.

Each item is described by keywords and values. A keyword and its associated value must be separated by an equal sign. Each keyword/value pair must be separated by at least one space. The recognized keywords are:

level A number that indicates the nesting level of the item. The first item must be level 0 (zero). Each subsequent nesting level must add 1 (one). The nesting level can be increased only when specifying a popup submenu.

This keyword must be provided.

id The user command number to be associated with the item. The id is the number that will be reported for the UserCommand event when the user chooses that menu item.

Two special ids are defined:

- If id is specified as ARS_OLE_MENU_ID_SEPARATOR, a separator item is created.
- If id is specified as ARS_OLE_MENU_ID_POPUP, a popup submenu is created. In this case, the following item should have a level one greater than the level specified for this item.

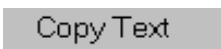
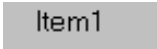
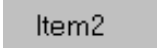
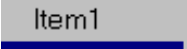
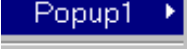
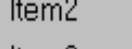
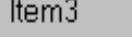
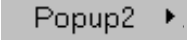

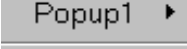


Other ids must have a minimum of ARS_OLE_MIN_MENU_ID and a maximum value of ARS_OLE_MAX_MENU_ID.

This keyword must be provided.

- enabled** The enabled keyword must have a value of 1 (one) or 0 (zero). If 1 (one), the item is enabled; if 0 (zero), disabled.
- This keyword is optional and is ignored for separator items. The default is for the item to be enabled.
- checked** The checked keyword must have a value of 1 (one) or 0 (zero). If 1 (one), the item is checked; if 0 (zero), unchecked.
- This keyword is optional and is ignored for separator items. The default is for the item to be unchecked.
- text** The text keyword specifies the text of the item. The text, which may include embedded blanks, must be contained in single quotes. If a single quote is part of the text, two consecutive single quotes must be specified.
- The text may contain the normal Windows special characters, such as an & (ampersand) to indicate that the following character is to be underlined.
- This keyword is optional and is ignored for separator items.

If the menu data is not valid, an error position is returned via the **pErrorPosition** parameter. The position is zero-based and is relative to the first character of the menu data. The position will normally identify the first character of the item which contains an error.

The following examples show the use of the keywords.

Data	Menu
level=0 id=368 text='Copy Text'	
level=0 id=44 text='Item1'\n	
level=0 id=1\n	
level=0 id=45 text='Item2'	
level=0 id=32457 text='Item1'\n	
level=0 id=0 text='Popup1'\n	
level=1 id=32458 text='Item2'\n	
level=1 id=32459 text='Item3'\n	
level=0 id=1\n	
level=0 id=0 text='Popup2'\n	
level=1 id=32460 enabled=0 text='Item4'\n	
level=1 id=1\n	
level=1 id=32461 checked=1 text='Item5'	
	
	

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: UserCommand event

Example:

The following example shows how to create a right button menu that contains the command Copy Text.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
VARIANT var;  
.  
.  
.  
  
rc = pArsCtrl->SetRightButtonMenu( "level=0 id=368 text='Copy Text'", var );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
Dim var As Variant  
  
.  
.  
  
rc = ArsOle.SetRightButtonMenu ("level=0 id=368 text='Copy Text'", var)  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If
```

SetSelectionMode

short **SetSelectionMode**(
 short **Mode**)

Parameters: **Mode**

Specifies the new selection mode. This must be one of the following selection mode values found in ARSOLEEX.H:

```
ARS_OLE_SELECTION_MODE_NONE  
ARS_OLE_SELECTION_MODE_AREA  
ARS_OLE_SELECTION_MODE_TEXT
```

ARS_OLE_SELECTION_MODE_NONE specifies that the user cannot select a portion of the document with the mouse. This is the default mode when a document is opened. Setting this mode removes any existing selection.

ARS_OLE_SELECTION_MODE_AREA specifies that user selection is performed in the same manner as Options->Selection Mode->Area with the OnDemand client GUI.

ARS_OLE_SELECTION_MODE_TEXT specifies that user selection is performed in the same manner as Options->Selection Mode->Text with the OnDemand client GUI.

Description: Subsequent mouse selection by the user is performed in the specified mode.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: CopyBitmap, CopyText

Example: The following example sets the selection mode to area selection.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
.  
.  
rc = pArsCtrl->SetSelectionMode( ARS_OLE_SELECTION_MODE_AREA );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
.  
.
```

Visual Basic Example

```
Dim rc As Integer

.
.

rc = ArsOle.SetSelectionMode (ARS_OLE_SELECTION_MODE_AREA)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.
```

SetServerPrinterData

short **SetServerPrinterData**(
 short **Index**,
 char * **pData**)

Parameters: **Index**

Specifies the server printer data index. It must be one of the following values found in ARSLOEEEX.H:

ARS_OLE_SERVER_PRINTER_DATA_FAX_RECEIVER_NAME
ARS_OLE_SERVER_PRINTER_DATA_FAX_RECEIVER_COMPANY
ARS_OLE_SERVER_PRINTER_DATA_FAX_RECEIVER_FAX_NUMBER
ARS_OLE_SERVER_PRINTER_DATA_FAX_SENDER_NAME
ARS_OLE_SERVER_PRINTER_DATA_FAX_SENDER_COMPANY
ARS_OLE_SERVER_PRINTER_DATA_FAX_SENDER_TEL_NUMBER
ARS_OLE_SERVER_PRINTER_DATA_FAX_SENDER_FAX_NUMBER
ARS_OLE_SERVER_PRINTER_DATA_FAX_SENDER_COVER_PAGE
ARS_OLE_SERVER_PRINTER_DATA_FAX_SUBJECT
ARS_OLE_SERVER_PRINTER_DATA_FAX_NOTES
ARS_OLE_SERVER_PRINTER_DATA_INFO_FROM
ARS_OLE_SERVER_PRINTER_DATA_INFO_TO

pData

Points to a null-terminated character string containing the data to be associated with the index. This parameter may be null.

Description: The data associated with each index defaults to the empty string. Any data set using this method is retained until changed. If pData is null, the data is reset to the empty string. When the PrintDoc method is used and a server printer is specified, the type of printer (such as FAX) is determined and the appropriate data is sent along with the document.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: PrintDoc

Example

The following example sets the FAX receiver name for server printers.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
  
.  
.  
.  
  
rc = pArsCtrl->SetServerPrinterData(  
    ARS_OLE_SERVER_PRINTER_DATA_FAX_RECEIVER_NAME,  
    "John Doe" );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
.  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
  
rc = ArsOle.SetServerPrinterData (   
    ARS_OLE_SERVER_PRINTER_DATA_FAX_RECEIVER_NAME,  
    "John Doe");  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

SetUserMessageMode

short **SetUserMessageMode**(
 short **Mode**)

Parameters: **Mode**

Specifies the new user message mode. This must be one of the following message mode values found in ARSOLEEX.H:

 ARS_OLE_USER_MSG_MODE_SHOW
 ARS_OLE_USER_MSG_MODE_SUPPRESS

ARS_OLE_USER_MSG_MODE_SHOW indicates that all OnDemand exception messages resulting from OLE Control operations are to be displayed to the user. ARS_OLE_USER_MSG_MODE_SUPPRESS indicates that all OnDemand exception messages resulting from OLE Control operations are to be suppressed.

Description: Subsequent OnDemand exception messages resulting from OLE Control operations are displayed or suppressed as requested.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

Example: The following example suppresses OnDemand exception messages resulting from OLE Control operations.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
.  
.  
rc = pArsCtrl->SetUserMessageMode( ARS_OLE_USER_MSG_MODE_SUPPRESS );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
  
rc = ArsOle.SetUserMessageMode (ARS_OLE_USER_MSG_MODE_SUPPRESS)  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

ShowFolder

```
short ShowFolder(  
    boolean Show,  
    short Left,  
    short Top,  
    short Right,  
    short Bottom )
```

Parameters:

Show

If non-zero, indicates that the OnDemand Folder dialog box is to be displayed; otherwise, that it is to be hidden.

Left

Specifies the X coordinate of the upper left point at which the dialog box is to be displayed. If **Show** is zero, this parameter is ignored. If **Show** is non-zero and this parameter is less than zero, the dialog box is displayed at its current size and location.

Top

Specifies the Y coordinate of the upper left point at which the dialog box is to be displayed.

Right

Specifies the X coordinate of the lower right point at which the dialog box is to be displayed.

Bottom

Specifies the Y coordinate of the lower right point at which the dialog box is to be displayed.

Description: The OnDemand Folder dialog box is displayed or hidden.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: OpenFolder, ActivateFolder

Example: The following example shows the OnDemand Folder dialog box centered in the screen with a border equal to 10% of the screen size.

C/C + + Example

```

CArsOle * pArsCtrl;
short rc, left, top, right, bottom;
int screen_width, screen_height;
.
.
screen_width = GetSystemMetrics( SM_CXSCREEN );
screen_height = GetSystemMetrics( SM_CYSCREEN );
left = screen_width / 10;
top = screen_height / 10;
right = screen_width * 8 / 10;
bottom = screen_height * 8 / 10;

rc = pArsCtrl->ShowFolder( TRUE, left, top, right, bottom );
if ( rc != ARS_OLE_RC_SUCCESS )
    ERROR;
.
.

```

Visual Basic Example

```

Dim rc As Integer

.
.

rc = ArsOle.ShowFolder (True, _
                        Screen.Width / 10, _
                        Screen.Height / 10, _
                        Screen.Width * 8 / 10, _
                        Screen.Height * 8 / 10)

If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.

```

ShowWaitCursorDuringCancelableOperation

short **ShowWaitCursorDuringCancelableOperation**(
boolean **Show**)

Parameters: **Show**

If zero, indicates that OnDemand should not display a wait cursor during SearchFolder or OpenDoc operations. If non-zero, indicates that a wait cursor should be displayed. This is the default behavior.

Description: During cancelable operations, OnDemand normally displays a wait cursor. If an application provides a cancel button or similar facility to allow the user to initiate a cancel, it may be desirable to prevent this display. This allows the application to show its own cursor, such as an hourglass and arrow, to tell the user that a cancel may be requested.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: CancelOperation

Example: The following example causes a wait cursor not to be displayed during a cancelable operation.

C/C++ Example

```
CArs01e * pArsCtrl;  
.  
.  
.  
pArsCtrl->ShowWaitCursorDuringCancelableOperation( FALSE );  
.  
.  
.
```

Visual Basic Example

```
.  
.  
.  
Ars01e.ShowWaitCursorDuringCancelableOperation (False)  
.  
.  
.
```

StoreDoc

```
short StoreDoc(  
    char *    DocPath,  
    char *    ApplGrpName,  
    char *    ApplName,  
    Variant * Values )
```

Parameters: DocPath

Specifies the fully-qualified path of a file containing the document data to be stored in the OnDemand database. This parameter is required.

ApplGrpName

Specifies the name of an Application Group within the active folder. It is the responsibility of the caller to know the Application Group names associated with the active folder. This parameter is required.

ApplName

Specifies the name of an Application within the specified Application Group. It is the responsibility of the caller to know the Application names associated with the specified Application Group. This parameter is required.

Values

Points to a SafeArray of folder values. Each value is a character string which will be converted to data of the field type (i.e., integer, date, etc.).

The number and order of folder fields can be determined by using the GetFolderFieldName(s) methods. For more information on these methods, see "GetFolderFieldName" on page 60 and "GetFolderFieldNames" on page 61.

Any folder fields not specified are given an empty string for string fields or zero for numeric fields. If extraneous fields are specified, they are ignored.

Date fields must be provided in the format required for the field (for example, specifying 02/03/96 is invalid when February 3, 1996 is required).

Description: OnDemand converts the folder field values to application group fields and stores the data from the specified file in the database as a document associated with the specified Application Group and Application.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetNumFolderFields, GetFolderFieldName, GetFolderFieldNames

Example:

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
SAFEARRAY * pSA;  
VARIANT var;  
BSTR bstrElement;  
long i;  
  
.  
.  
  
pSA = SafeArrayCreateVector(VT_BSTR, 0, 2);  
if ( pSA == NULL )  
    ERROR;  
  
bstrElement = SysAllocStringByteLen ("255-546-667", 11);  
i = 0;  
SafeArrayPutElement (pSA, &i, bstrElement);  
bstrElement = SysAllocStringByteLen ("06/07/94", 8);  
i = 1;  
SafeArrayPutElement (pSA, &i, bstrElement);  
  
var.vt = VT_ARRAY | VT_BSTR;  
var.parray = pSA;  
  
rc = pArsCtrl->StoreDoc( "g:\\download\\file.afp",  
                        "BKH-CRD",  
                        "BKH-CRD",  
                        &var );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
.  
.
```

Visual Basic Example

```
Dim values(2) As String
Dim rc As Integer

.
.

values(0) = "255-546-667"
values(1) = "06/07/94"
var = values

rc = ArsOle.StoreDoc ("g:\download\file.afp", _
                    "BKH-CRD", _
                    "BKH-CRD", _
                    var)
if rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.
```

UpdateDoc

```
short UpdateDoc(  
    long *    DocIndex,  
    char *    pFieldName,  
    char *    pValue,  
    ...  
);
```

Parameters: **DocIndex**

Specifies zero-based relative document number within the document list of the active folder. This parameter is required.

pFieldName

Specifies the name of a folder field. This parameter is required.

pValue

Specifies the value to be stored in the specified folder field. This value is a character string which will be converted to data of the field type (i.e., integer, date, etc.).

Date fields must be provided in the format required for the field (for example, specifying 02/03/96 is invalid when February 3, 1996 is required).

This parameter is required.

Description: OnDemand converts the folder field value to an application group field and updates the data from the specified value in the database.

Return Value: Refer to "Return Code" on page 4 for an explanation of the return code.

See Also: GetNumFolderFields, GetFolderFieldName, GetFolderFieldNames

Example: The following example updates the "Balance" field of the first document within the document list of the active folder.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
  
.  
.  
  
rc = pArsCtrl->UpdateDoc( 0,  
                           "Balance",  
                           "123.45" );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
.  
.
```

Visual Basic Example

```
Dim rc As Integer

.
.

rc = ArsOle.UpdateDoc ( 0,
                        "Balance", -
                        "123.45" )
if rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

.
.
```

UndoFind

short **UndoFind**()

Description: Highlighting is removed from the current found string.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: FindStringInDoc

Example: The following example removes highlighting from a found string.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
  
.  
.  
rc = pArsCtrl->UndoFind( );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
.  
.
```

Visual Basic Example

```
Dim rc As Integer  
  
.  
.  
  
rc = ArsOle.UndoFind ()  
If rc <> ARS_OLE_RC_SUCCESS Then  
    MsgBox "ERROR"  
End  
End If  
  
.  
.
```

WasOperationCancelled

short **WasOperationCancelled**(
 VARIANT * **pCancelled**)

Parameters: **pCancelled**

Points to a variable to receive the result. On return, this variable is set to type VT_I2.

Description: If the most recent SearchFolder, OpenDoc, or RetrieveDoc operation was cancelled, the result variable is set to a non-zero value; otherwise, to zero.

Return Value: Refer to “Return Code” on page 4 for an explanation of the return code.

See Also: CancelOperation

Example: The following example searches a folder and determines whether the search was cancelled.

C/C++ Example

```
CArsOle * pArsCtrl;  
short rc;  
VARIANT cancelled;  
  
.  
.  
  
rc = pArsCtrl->SearchFolder( FALSE );  
if ( rc != ARS_OLE_RC_SUCCESS )  
    ERROR;  
  
rc = pArsCtrl->WasOperationCancelled( &cancelled );  
if ( cancelled.iVAL )  
    CANCELLATION LOGIC;  
  
.  
.
```

Visual Basic Example

```

Dim rc As Integer
Dim cancelled As Variant

.
.

rc = ArsOle.SearchFolder (False)
If rc <> ARS_OLE_RC_SUCCESS Then
    MsgBox "ERROR"
End
End If

rc = ArsOle.WasOperationCancelled (cancelled)
If cancelled <> 0 Then
    CANCELLATION LOGIC
End If

.
.

```

OLE Events

The following events are fired by an OnDemand OLE Control.

FolderSearchCompleted

This event is fired when a folder search has completed. The search may have been initiated

- By the container application invoking the SearchFolder method; or
- By the user if the ShowFolder method was used to display the OnDemand Folder dialog box.

FolderClosed

This event is fired when a folder is closed. The close may have been initiated

- By the container application invoking the CloseFolder method;
- By the user if the ShowFolder method was used to display the OnDemand Folder dialog box;
- By a logoff; or
- By destruction of the control.

DocOpened

This event is fired when a document is opened. The open may have been initiated

- By the container application invoking the OpenDoc method; or
- By the user if the ShowFolder method was used to display the OnDemand Folder dialog box.

DocClosed

This event is fired when a document is closed. The close may have been initiated

- By the container application invoking the CloseDoc method;
- By the folder associated with the document being closed;
- By a logoff; or
- By destruction of the control.

AreaSelected

This event is fired when the user selects an area of the screen. This is possible only if the SetSelectionMode method has been used to set the selection mode to ARS_OLE_SELECTION_MODE_AREA or ARS_OLE_SELECTION_MODE_TEXT.

AreaDeselected

This event is fired when a selection is removed. The removal may have been caused

- By the user double clicking the mouse outside of the selection;
- By the SetSelectionMode method specifying ARS_OLE_SELECTION_MODE_NONE;
- By the document being closed; or
- For certain documents, by changing the page, scroll position, zoom, or other presentation attributes.

UserCommand(long CommandID)

This event is fired when the user clicks the right mouse button and chooses a menu item. The command ID associated with the menu item is passed as a parameter.

See the SetRightButtonMenu method for information on creating menus.

Windows 32-bit GUI Customization Guide

OnDemand Customization Overview

The OnDemand application can be customized by specifying command line parameters, by invoking and manipulating it from another application with the Dynamic Data Exchange interface, by invoking other applications and DLLs from the client, by retrieving related documents, by creating a Product Information File, and by auditing documents.

Command Line

This section describes:

- Starting OnDemand.
- The parameter syntax rules used for the command line parameters.
- The parameters recognized by the OnDemand 32-bit Windows Client.

Starting OnDemand 32-bit client

OnDemand can be started by double-clicking on the program icon on your desktop.

Parameter Syntax

The following syntax rules apply to command line parameters:

- At least one space character must follow the executable name.
- A parameter is identified by a slash followed immediately by a single character.
- The parameter character is not case sensitive.
- The associated parameter value, if any, begins with the first non-space character following the parameter character and ends with the last non-space character preceding the next parameter or the end of the command line.
- If a slash character is to be included in the parameter value, two consecutive slash characters must be specified.
- Keyword parameter values are not case sensitive.
- Unrecognized parameters are ignored.
- If the same parameter appears more than once, the last occurrence is used.

Parameters

The following parameters are recognized by the OnDemand for Windows NT and Windows 95:

Product Title — /T name

If this parameter is specified, OnDemand replaces the default product title at the top of the main OnDemand window and on a number of menu items. This title takes precedence over a title supplied in a Product Information File (refer to “Product Information File” on page 273).

Example: C:\ARS32\ARSGUI32 /T Joe's Windows Application

Logon Server Name — /S name

If this parameter is specified, OnDemand uses the value as the server name for the initial user logon. The name must be one of the server names defined in the Registry. (Use the Update Servers command to add server names to the Registry.)

This parameter may be used in combination with the Logon User ID and Logon Password parameters to logon the user during initialization. If all three parameters are specified and are valid, the initial Logon dialog box is not displayed.

Example: C:\ARS32\ARSGUI32 /S pikes /U user1 /P pass1

Logon User ID — /U id

If this parameter is specified, OnDemand uses the value as the user id for the initial user logon.

This parameter may be used in combination with the Logon Server Name and Logon Password parameters to logon the user during initialization. If all three parameters are specified and are valid, the initial Logon dialog box is not displayed.

Example: C:\ARS32\ARSGUI32 /S pikes /U user1 /P pass1

Logon Password — /P password

If this parameter is specified, OnDemand uses the value as the password for the initial user logon.

This parameter may be used in combination with the Logon Server Name and Logon User ID parameters to logon the user during initialization. If all three parameters are specified and are valid, the initial Logon dialog box is not displayed.

Example: C:\ARS32\ARSGUI32 /S pikes /U user1 /P pass1

Change Password — /C new password

If this parameter is specified, OnDemand changes the password for the user after a successful logon. This parameter can only be specified if the /S, /U, and /P parameters have also been specified.

Example: C:\ARS32\ARSGUI32 /S pikes /U user1 /P pass1 /C newpass

Folder Name — /F name

If this parameter is specified, OnDemand uses the value as the folder name of the initial folder to be opened. If the value is a valid folder name, the initial Open a Folder dialog box is not displayed.

This parameter may be used in combination with the Logon Server Name, Logon User ID, Logon Password, Disable Logoff and Password Change, and Maximum Open Folders parameters to limit a user to operations with a single folder on a single server.

Example: C:\ARS32\ARSGUI32 /F Credit Card Statements

Maximum Open Folders — /O number

If this parameter is specified, OnDemand limits the number of open folders to no more than the indicated value. This parameter may be used in combination with the Free Memory When Folder Closed parameter to cause folder information to be refreshed from the server each time that a folder is opened.

Example: C:\ARS32\ARSGUI32 /O 1 /Q

Window Placement — /W placement

If this parameter is specified, OnDemand displays the main window as indicated by the placement. The placement must be one of the following:

- **Z** to indicate that the main window is to be *zoomed* (maximized).
- **I** to indicate that the main window is to be *iconized* (minimized).
- **N** to indicate that the main window is *not* to be displayed (made invisible). This option should be specified only if OnDemand is to be manipulated with the DDE interface.
- **x,y,w,h** to indicate the main window placement, where
 - **x** is the left origin in percent of the screen width.
 - **y** is the top origin in percent of the screen height.
 - **w** is the width in percent of the screen width.
 - **h** is the height in percent of the screen height.

If this parameter is not specified or if no placement value is specified, the main window appears as it did when OnDemand was last terminated.

Example: C:\ARS32\ARSGUI32 /W 5,10,90,80

Enable DDE Interface — /I number,path,resid

If this parameter is specified, OnDemand enables the Dynamic Data Exchange interface and prepares to accept commands from another application.

number, **path**, and **resid** are interpreted as follows:

- **number** must be an integer between 1 and 5. It indicates the number of DDE-application switch menu items to be provided (for more information, see “ENABLE_SWITCH” on page 215). The default value is 1.
- **path** specifies the fully-qualified name of a resource DLL containing a bitmap for toolbar buttons to be associated with the DDE-application switch menu items. The size of the bitmap depends on the number of menu items. Each button requires a 16 X 15 pel image. If there is a single menu item, the bitmap should be 16 X 15; if two menu items, 16 X 30; and so forth.
- **resid** specifies the bitmap resource identifier within the DLL. It must be an integer value.

Example C:\ARS\ARSGUI32 /I 3,C:\MYAPP\BITMAP.DLL,81

Disable Exit — /K

If this parameter is specified, OnDemand disables the Exit menu item.

Example: C:\ARS\ARSGUI32 /K

Disable Logoff or Password Change — /X

Use the **X** parameter to indicate if OnDemand disables logoff or a change to the logon password as follows:

- If **X** is specified with no options, OnDemand disables the Logoff and Change Logon Password menu items.
- If **X** is specified with the **L** option, OnDemand disables the Logoff menu item.
- If **X** is specified with the **P** option, OnDemand disables the Change Logon Password menu item.

Example: C:\ARS32\ARSGUI32 /X

Disable Update Servers — /Y

If this parameter is specified, OnDemand disables the Update Servers button on the Logon dialog box.

Example: C:\ARS32\ARSGUI32 /Y

Disable Close Folder — /Z

If this parameter is specified, OnDemand disables the Close Folder menu item, the Close Folder button on the Search Criteria and Document List dialog box, and the Close menu item on the system menu of the Search Criteria and Document List dialog box.

Example: C:\ARS32\ARSGUI32 /Z

Disable Anticipation — /V

If this parameter is specified, OnDemand does not attempt to anticipate the next user request, such as displaying the Logon dialog box after initialization or displaying the Open Folder dialog box after the current folder is closed.

Example: C:\ARS32\ARSGUI32 /V

Disable User Confirmation — /B

If this parameter is specified, OnDemand does not request confirmation of the user's action when there is an open document during folder close, logoff, or exit.

Example: C:\ARS32\ARSGUI32 /B

Free Memory When Folder Closed — /Q

If this parameter is specified, OnDemand frees all memory associated with a folder when it is closed and refreshes the folder information from the server when it is reopened.

Example: C:\ARS32\ARSGUI32 /Q

Language Path — /I

This parameter identifies the full path to the files that support the national language environment of the client.

Example: `C:\ARS32\ARSGUI32 /1 C:\ARS32\ARSGUI32\LOCALE\ENU`

Dynamic Data Exchange (DDE) and DDE Management Library

Dynamic Data Exchange (DDE) is an interprocess communication mechanism supported by Windows. Two Windows applications carry on a “conversation” with DDE protocols. OnDemand can function as a server in such a conversation, providing access to data and performing actions at the direction of a client application. The client application is able to manipulate OnDemand by “remote control.”

DDE is based on the messaging system built into Windows. The applications communicate by posting messages to each other. This original DDE communication protocol can be difficult to understand and implement. Windows has subsequently hidden many of the complexities of DDE by providing the DDE Management Library (DDEML), which is a function call interface.

Either the original DDE protocol or the DDEML can be used by a client application when communicating with OnDemand. The description and examples in the following sections use the DDEML, and assume a general familiarity with its operation. Refer to the *Microsoft Windows Software Development Kit* or an appropriate textbook for a more complete description of DDE and DDEML.

Invoking OnDemand 32-bit from Another Windows Application

This invocation method is typically used when the Dynamic Data Exchange interface is required.

OnDemand 32-bit can be invoked from another Windows 95/NT application by using the *CreateProcess* Windows API. The prototype for this function is

```
BOOL CreateProcess(  
    lpzImageName,  
    lpzCmdLine,  
    lpsaProcess,  
    lpsaThread,  
    fInheritHandles,  
    fdwCreate,  
    lpvEnvironment,  
    lpzCurDir,  
    lpsiStartInfo,  
    lppiProcInfo )
```

lpzCmdLine is a null-terminated character string containing the name of the OnDemand executable followed by a set of optional parameters as described below.

Consult your documentation for a description of the other parameters.

This invocation method is typically used when the Dynamic Data Exchange interface is required.

The following is an example of invocation using CreateProcess:

```
PROCESS_INFORMATION pi;
STARTUPINFO sui;
char * pCmd;

memset( &sui, 0, sizeof(STARTUPINFO) );
sui.cb = sizeof(STARTUPINFO);

pCmd = "C:\\\\ARS\\\\ARSGUI32 "
       "/T Special OnDemand "
       "/I 3,C:\\\\MYAPP\\\\BITMAP.DLL,81 "
       "/W 5,10,90,80";

CreateProcess( NULL,
              pCmd,
              NULL,
              NULL,
              FALSE,
              CREATE_NEW_CONSOLE,
              NULL,
              NULL,
              &sui,
              &pi );
```

OnDemand Invocation and DDEML Initialization

OnDemand must be an active Windows application before a DDE conversation between it and a client application can be established. The client application typically invokes OnDemand as described in “Invoking OnDemand 32-bit from Another Windows Application” on page 191, specifying (at a minimum) the Enable DDE Interface parameter. This parameter causes OnDemand to enable its DDE interface; if it is not specified, OnDemand ignores all DDE communication.

Other parameters can be used to position the window, provide a custom title, logon a user, open a folder, etc. Many of these actions can also be performed by DDE functions. The command line parameters should be used to establish the initial appearance of the OnDemand window. Refer to “Parameters” on page 185 for a complete description of the command line parameters.

After OnDemand has been made active, the client application must identify itself to the DDEML. This is done with the DdeInitialize DDEML function.

Finally, the client application must establish a DDE conversation with OnDemand. This is done with the DdeConnect DDEML function. The service name of the OnDemand server application is ARS. The topic name for the conversation is also ARS.

The following is a typical example for establishing a DDE conversation with OnDemand for Windows.

```

/* Global Variables */
DWORD DdeInstance;
HCONV hDdeConv;
.
.
.
/* Local Variables */
FARPROC pfnDdeCallBack;
HSZ hDdeString1, hDdeString2;
UINT rc;
char cmdline[500], buffer[500];
.
.
.
/* Invoke OnDemand */
PROCESS_INFORMATION pi;
STARTUPINFO sui;

memset( &sui, 0, sizeof(STARTUPINFO) );
sui.cb = sizeof(STARTUPINFO);

rc = CreateProcess( NULL,
                   cmdline,
                   NULL,
                   NULL,
                   FALSE,
                   CREATE_NEW_CONSOLE,
                   NULL,
                   NULL,
                   &sui,
                   &pi );

if ( !rc )
{
    sprintf( buffer,
            "CreateProcess of '%s' failed with error %ld",
            cmdline,
            (long)GetLastError( ) );
    MESSAGE( buffer );
    return;
}

```

```

/* Initialize DDEML */
pfnDdeCallBack = MakeProcInstance( (FARPROC)DdeCallBack, hInst );
DdeInstance = 0;
DdeInitialize( &DdeInstance,
               (PFNCALLBACK)pfnDdeCallBack,
               APPCLASS_STANDARD | APPCMD_CLIENTONLY,
               0L );
if ( DdeInstance == 0 )
{
    MESSAGE( "DdeInitialize failed" );
    return;
}
/* Connect to OnDemand DDE Interface */
hDdeString1 = DdeCreateStringHandle( DdeInstance, "ARS", 0 );
hDdeString2 = DdeCreateStringHandle( DdeInstance, "ARS", 0 );
for ( j = 0; j < 100; j++ )
{
    hDdeConv = DdeConnect( DdeInstance, hDdeString1, hDdeString2, NULL );
    if ( hDdeConv != NULL )
        break;
}
DdeFreeStringHandle( DdeInstance, hDdeString1 );
DdeFreeStringHandle( DdeInstance, hDdeString2 );
if ( hDdeConv == NULL )
{
    MESSAGE( "Unable to connect to OnDemand" );
    return;
}
.
.
.

```

DDEML Termination

When the client application wishes to terminate the DDE conversation, it simply uses the `DdeUninitialize` DDEML function. This informs `OnDemand` that its DDE client has ended the conversation, but it does not cause `OnDemand` to terminate. If the client wishes `OnDemand` to terminate, it should tell `OnDemand` to `EXIT` (refer to “EXIT” on page 216) before ending the conversation.

The client must also be prepared to have the DDE conversation ended by `OnDemand`. If `OnDemand` is terminated, either independently or as a result of a DDE `EXIT`, the client receives a `XTYP_DISCONNECT` transaction from the DDEML.

DDEML Transactions

All OnDemand commands are DDEML XTYP_REQUEST transactions. Some cause an operation to be performed; some return additional data; and all provide a return code.

The DDEML DdeClientTransaction function is used to initiate the transaction. The DDEML item name string contains the command and concatenated parameters. The syntax is the same as the command line (refer to “Parameter Syntax” on page 185) with the DDE command replacing the executable name.

The DDEML DdeClientTransaction function returns a data handle containing a return code, and optionally, additional data. The return code is a set of ASCII digits representing one of the values described for the individual commands and summarized in “Return Codes” on page 256. These ASCII digits are always followed by a single space character. If additional data is present, it is a null-terminated character string beginning at the character immediately following the space character.

The following example describes a C function that executes OnDemand DDE commands and receives return information. All the examples for the individual OnDemand DDE commands are based on this C function example.

```
/* Global Variables */
DWORD DdeInstance;
HCONV hDdeConv;

#define ERROR_MAP struct _ErrorMap
ERROR_MAP
{
    int    code;
    char * pMsg;
};

static ERROR_MAP Errors[] =
{
    { ARS_DDE_RC_UNKNOWN_COMMAND,    "Unknown command." },
    { ARS_DDE_RC_PARM_NOT_SPECIFIED,  "Parameter not specified." },
    { ARS_DDE_RC_INVALID_PARM_VALUE,  "Invalid parameter value." },
    { ARS_DDE_RC_SERVER_ERROR,        "Server error." },
    { ARS_DDE_RC_FILE_ERROR,          "File error." },
    { ARS_DDE_RC_NOT_LOGGED_ON,        "Not logged on." },
    { ARS_DDE_RC_MAX_FOLDERS_OPEN,     "Maximum folders open." },
    { ARS_DDE_RC_FOLDER_NOT_OPEN,      "Folder not open." },
    { ARS_DDE_RC_NO_DOC,               "No document exists." },
    { ARS_DDE_RC_OPEN_DOCS,            "Documents are open." },
};
#define NUM_ERRORS ( sizeof(Errors) / sizeof(ERROR_MAP) )

.
.
.
```

```

BOOL DoDdeCommand( char * pCmd,      /* -> Command string      */
                  char * pParms,    /* -> Command parameters  */
                  char * pData ) /* -> Buffer for returned data */
{
    HSZ hDdeString;
    HDDEDATA hDdeResult;
    DWORD data_len;
    char * pString;
    int j, rc;

    /* Add parameters to command line. */
    if ( pParms == NULL )
        pParms = "";
    pString = malloc( strlen( pCmd ) + strlen( pParms ) + 2 );
    strcpy( pString, pCmd );
    strcat( pString, " " );
    strcat( pString, pParms );

    /* Perform OnDemand DDE command. */
    hDdeString = DdeCreateStringHandle( DdeInstance, pCmd, 0 );
    hDdeResult = DdeClientTransaction( NULL,
                                       0,
                                       hDdeConv,
                                       hDdeString1,
                                       CF_TEXT,
                                       type,
                                       10000L,
                                       NULL );
    DdeFreeStringHandle( DdeInstance, hDdeString );
    free( pString );
}

```

```

/* Process command result. */
if ( hDdeResult == NULL )
{
    /* This should never happen. */
    MESSAGE( "DDE Timeout." );
    return FALSE;
}
else
{
    pString = (char*)DdeAccessData( hDdeResult, &data_len );
    rc = atoi( pString );
    if ( rc == ARS_DDE_RC_NO_ERROR )
    {
        if ( pData != NULL )
            strcpy( pData, strchr( pString, ' ' ) + 1 );
    }
    else
    {
        if ( pData != NULL )
            pData[0] = '\0';
        for ( j = 0; j < NUM_ERRORS; j++ )
            if ( Errors[j].code == rc )
                break;
        MESSAGE( j < NUM_ERRORS ? Errors[j].pMsg : "Error - invalid return code." );
    }
    DdeUnaccessData( hDdeResult );
    return rc == ARS_DDE_RC_NO_ERROR;
}
}

```

OnDemand DDE Commands

A header file, ARSDDEEX.H, is distributed with OnDemand. The header file contains symbolic definitions for many of the values used with the DDE commands. The header file can be included in C/C++ implementations or used as a reference for other languages.

This header file is installed into the INC subdirectory of the OnDemand installation directory. This subdirectory can be added to the include file path or the file can be copied to another directory.

ACTIVATE_DOC

Command	Parameters
ACTIVATE_DOC	<i>/D doc id</i>

Parameters: **D**

Specifies the document identifier, returned by the OPEN_DOC command, of the document to be activated. This parameter is required.

Action: OnDemand makes the document the active document by bringing its document window to the top and giving it the input focus.

Return Code:

- 0** ARS_DDE_RC_NO_ERROR
- 2** ARS_DDE_RC_PARM_NOT_SPECIFIED
- 3** ARS_DDE_RC_INVALID_PARM_VALUE
- 6** ARS_DDE_RC_NOT_LOGGED_ON
- 11** ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "ACTIVATE_DOC", "/D 1234", NULL );
```

ACTIVATE_FOLDER

Command	Parameters
ACTIVATE_FOLDER	<i>/F folder name</i>

Parameters: F

Specifies the folder name of a currently open folder. This parameter is required.

Action: OnDemand makes the specified folder the active folder.

Return Code:

- 0 ARS_DDE_RC_NO_ERROR
- 2 ARS_DDE_RC_PARM_NOT_SPECIFIED
- 3 ARS_DDE_RC_INVALID_PARM_VALUE
- 6 ARS_DDE_RC_NOT_LOGGED_ON
- 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "ACTIVATE_FOLDER", "Mary's Folder", NULL );
```

ANNOTATE_DOC

Command	Parameters
ANNOTATE_DOC	<i>/N doc number /P annotation path /G page number /U /C</i>

Parameters: N

Specifies the zero-based relative document number within the document list of the active folder. The number of documents in the list can be determined by using the GET_NUM_DOCS_IN_LIST command (refer to “GET_NUM_DOCS_IN_LIST” on page 222). The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command (refer to “GET_DOC_VALUES” on page 218).

This parameter is required.

P

Specifies the fully-qualified path of a file containing the text of the annotation. If the file contains more than 32,700 bytes, the text is truncated.

This parameter is required.

G

Specifies the document page number associated with the annotation.

This parameter is optional.

U

Indicates that the annotation is public rather than private.

This parameter is optional.

C

Indicates that the annotation may be copied to other servers.

This parameter is optional.

Action: OnDemand adds the annotation to the specified document.

Return Code:

- 0 ARS_DDE_RC_NO_ERROR
- 2 ARS_DDE_RC_PARM_NOT_SPECIFIED
- 4 ARS_DDE_RC_SERVER_ERROR
- 5 ARS_DDE_RC_FILE_ERROR
- 8 ARS_DDE_RC_FOLDER_NOT_OPEN

- 9 ARS_DDE_RC_NO_DOC
- 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS
- 12 ARS_DDE_RC_UNAUTHORIZED_OPERATION

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
char parms[200];

sprintf( parms,
        "/N %s /L %s /C %s /R /K",
        "22",
        "HP LaserJet on LPT1.AES:",
        "2" );

DoDdeCommand( "PRINT_DOC", parms, NULL );
```


ARRANGE_DOCS

Command	Parameters
ARRANGE_DOCS	/C /H /V /Z /R

Parameters: **C**

Indicates that the document windows are to be cascaded.

H

Indicates that the document windows are to be tiled horizontally.

V

Indicates that the document windows are to be tiled vertically.

Z

Indicates that the document windows are to be zoomed (maximized).

R

Indicates that the document windows are to be restored. This is the default parameter.

Action: OnDemand arranges the document windows as indicated by the parameter. If no parameter is specified, R is assumed. If multiple parameters are specified, the results are unpredictable.

Return Code:

0 ARS_DDE_RC_NO_ERROR
2 ARS_DDE_RC_PARM_NOT_SPECIFIED
6 ARS_DDE_RC_NOT_LOGGED_ON
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "ARRANGE_DOCS", "/V", NULL );
```

CHANGE_PASSWORD

Command	Parameters
CHANGE_PASSWORD	<i>/C Current Password /N New Password /V New Password</i>

Parameters: **C**

Specifies the users current password.

N

Specifies users new password.

V

Specifies the users new password again; this is for verification.

Action: OnDemand changes the logon password for the active user.

Return Code:

0 ARS_DDE_RC_NO_ERROR
2 ARS_DDE_RC_PARM_NOT_SPECIFIED
4 ARS_DDE_RC_SERVER_ERROR
6 ARS_DDE_RC_NOT_LOGGED_ON
22 ARS_DDE_RC_INCORRECT_CURRENT_PASSWORD
23 ARS_DDE_RC_PASSWORD_TOO_SHORT
24 ARS_DDE_RC_NEW_PASSWORD_MISMATCH

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "CHANGE_PASSWORD", "/C tt1sd /N sfd45r /V sfd45r", NULL );
```

CLEAR_FIELDS

Command	Parameters
CLEAR_FIELDS	None.

Parameters: None.

Action: OnDemand clears the search criteria entry windows for the active folder.

Return Code:

- 0** ARS_DDE_RC_NO_ERROR
- 8** ARS_DDE_RC_FOLDER_NOT_OPEN
- 11** ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "CLEAR_FIELDS", "", NULL );
```

CLOSE_ALL_DOCS

Command	Parameters
CLOSE_ALL_DOCS	None.

Parameters: None.

Action: OnDemand closes all open documents, destroys all document windows, and displays the active folder window.

Return Code:

- 0 ARS_DDE_RC_NO_ERROR
- 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "CLOSE_ALL_DOCS", "", NULL );
```

CLOSE_ALL_FOLDERS

Command	Parameters
CLOSE_ALL_FOLDERS	None.

Parameters: None.

Action: OnDemand closes all open folders.

Return Code:

- 0 ARS_DDE_RC_NO_ERROR
- 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "CLOSE_ALL_FOLDERS", "", NULL );
```

CLOSE_DOC

Command	Parameters
CLOSE_DOC	<i>/D doc id</i>

Parameters: **D**

Specifies the document identifier, returned by the OPEN_DOC command, of the document to be closed. This parameter is required.

Action: OnDemand closes the document by destroying the document window.

Return Code:

- 0** ARS_DDE_RC_NO_ERROR
- 2** ARS_DDE_RC_PARM_NOT_SPECIFIED
- 3** ARS_DDE_RC_INVALID_PARM_VALUE
- 6** ARS_DDE_RC_NOT_LOGGED_ON
- 11** ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "CLOSE_DOC", "/D 1234", NULL );
```

CLOSE_FOLDER

Command	Parameters
CLOSE_FOLDER	None.

Parameters: None.

Action: OnDemand closes the active folder. If there are other open folders, one of them becomes the active folder.

Return Code:

- 0** ARS_DDE_RC_NO_ERROR
- 8** ARS_DDE_RC_FOLDER_NOT_OPEN
- 11** ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "CLOSE_FOLDER", "", NULL );
```

COPY_DOC_PAGES

Command	Parameters
COPY_DOC_PAGES	<i>/P path /G page number C /A</i>

Parameters: **P**

Specifies the fully-qualified path of a file to which the pages are to be copied. If the file does not exist, it is created; if it does exist, the pages are appended to the existing file. This parameter is required.

G

Specifies the pages to be copied:

Specify the *page number* or *page numbers* for the pages you want to copy.

You can specify a single page or a range of pages. You can also specify a combination of single pages and ranges of pages separated by a comma. For example,

/G 3, 5-7, 9, 110-120

copies page 3, pages 5 through 7, page 9, and pages 110 to 120.

If the **G** parameter is not specified, all pages are copied.

When you specify the **G** parameter, you can specify the page or range of pages or the current page by specifying **C**, but you cannot specify pages and **C** at the same time.

The **G** parameter is optional.

C

Specify a **C** to copy the current page.

The **C** parameter is optional.

A

If this parameter is specified, the pages are copied “AS IS” in the data stream format; otherwise, the pages are copied as ASCII text (for AFP and line data only).

This parameter is optional.

Action: OnDemand copies the specified page(s) from the active document to the specified file. If a list of pages is specified, the pages are copied in the order you specify.

Return Code:

- 2 ARS_DDE_RC_PARM_NOT_SPECIFIED
- 3 ARS_DDE_RC_INVALID_PARM_VALUE
- 4 ARS_DDE_RC_SERVER_ERROR
- 5 ARS_DDE_RC_FILE_ERROR
- 6 ARS_DDE_RC_NOT_LOGGED_ON
- 12 ARS_DDE_RC_UNAUTHORIZED_OPERATION
- 13 ARS_DDE_RC_USER_CANCELLED_OPERATION
- 14 ARS_DDE_RC_NO_ACTIVE_DOC

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
char parms[200];

sprintf( parms,
        "/P %s /G 6, 3-4, 8, 112-118",
        "C:\\ASCII.TXT")

DoDdeCommand( "COPY_DOC_PAGES", parms, NULL );
```

DELETE_DOC

Command	Parameters
DELETE_DOC	<i>/N doc number</i>

Parameters: N

Specifies the zero-based relative document number within the document list of the active folder. The number of documents in the list can be determined by using the GET_NUM_DOCS_IN_LIST command (refer to “GET_NUM_DOCS_IN_LIST” on page 222). The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command (refer to “GET_DOC_VALUES” on page 218).

This parameter is required.

The *doc number* may be specified as –1 to indicate that all selected documents are to be deleted.

Action: OnDemand deletes the specified document or all selected documents from the database. The deleted document(s) are removed from the Document List. Since the document numbers may have changed, information from a previous GET_DOC_VALUES command may no longer be valid.

Return Code:

- 0 ARS_DDE_RC_NO_ERROR
- 2 ARS_DDE_RC_PARM_NOT_SPECIFIED
- 3 ARS_DDE_RC_INVALID_PARM
- 4 ARS_DDE_RC_SERVER_ERROR
- 8 ARS_DDE_RC_FOLDER_NOT_OPEN
- 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS
- 12 ARS_DDE_RC_UNAUTHORIZED_OPERATION

Return Data: OnDemand returns the number of documents that were successfully deleted. The returned null-terminated string can be converted to a long integer.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "DELETE_DOC", "23", NULL );
```

DESELECT_DOC

Command	Parameters
DESELECT_DOC	/N <i>doc number</i>

Parameters: N

Specifies the zero-based relative document number within the document list of the active folder. The number of documents in the list can be determined by using the GET_NUM_DOCS_IN_LIST command (refer to “GET_NUM_DOCS_IN_LIST” on page 222). The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command (refer to “GET_DOC_VALUES” on page 218).

This parameter is required.

The *doc number* may be specified as -1 to indicate that all documents are to be deselected.

Action: OnDemand deselects (removes highlight from) the Document List line that corresponds to the specific document number.

Return Code:

- 0 ARS_DDE_RC_NO_ERROR
- 2 ARS_DDE_RC_PARM_NOT_SPECIFIED
- 3 ARS_DDE_RC_INVALID_PARM_VALUE
- 8 ARS_DDE_RC_FOLDER_NOT_OPEN
- 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "DESELECT_DOC", "15", NULL );
```

DISABLE_SWITCH

Command	Parameters
DISABLE_SWITCH	/ N <i>number</i>

Parameters: N

Specifies the number of the menu item/toolbar button to be disabled. The number must be an integer between 1 and the number specified with the “Enable DDE Interface — /I number,path,resid” on page 187. Specifying 1 disables the first menu item/toolbar button; specifying 2 disables the second; and so forth.

This parameter is optional. The default is 1.

Action: OnDemand disables a toolbar icon and a menu item that allow the user to transfer window focus to the client application.

Return Code:

0 ARS_DDE_RC_NO_ERROR
3 ARS_DDE_RC_INVALID_PARM_VALUE
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "DISABLE_SWITCH", "", NULL );
```

ENABLE_SWITCH

Command	Parameters
ENABLE_SWITCH	<i>/H client hWnd /C client name /N number</i>

Parameters: H

Specifies, as a set of ASCII digits, the window handle of the client application. This parameter is required. The example below indicates how to prepare this parameter.

C

Specifies the name of the client application. This parameter is optional. If not specified, the name is not changed.

N

Specifies the number of the menu item/toolbar button to be enabled. The number must be an integer between 1 and the number specified with the “Enable DDE Interface — /I number,path,resid” on page 187. Specifying 1 enables the first menu item/toolbar button; specifying 2 enables the second; and so forth.

This parameter is optional. The default is 1.

Action: OnDemand enables a toolbar icon and a menu item that allow the user to transfer window focus to the client application. The client name is set as the menu item text.

If the client application has established an Advise Loop (see “DDEML Advise Loop” on page 258 for more information), OnDemand informs the client application when the user transfers focus.

Return Code:

- 0 ARS_DDE_RC_NO_ERROR
- 2 ARS_DDE_RC_PARM_NOT_SPECIFIED
- 3 ARS_DDE_RC_INVALID_PARM_VALUE
- 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
char parms[200];

sprintf( parms,
        "/H %ld /C %s",
        (long)hWnd,
        "Fred's Windows Application" );
DoDdeCommand( "ENABLE_SWITCH", parms, NULL );
```

EXIT

Command	Parameters
EXIT	None.

Parameters: None.

Action: OnDemand attempts to terminate when this command is received. If at least one document is open, and the Disable User Confirmation command line parameter has not been specified (refer to “Disable User Confirmation — /B” on page 188), a message box is displayed asking the user to confirm program exit.

If OnDemand terminates, the client receives a XTYP_DISCONNECT transaction from the DDEML.

Return Code:

0 ARS_DDE_RC_NO_ERROR
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "EXIT", "", NULL );
```

GET_DISPLAY_FIELDS

Command	Parameters
GET_DISPLAY_FIELDS	None.

Parameters: None.

Action: None.

Return Code:

0 ARS_DDE_RC_NO_ERROR
8 ARS_DDE_RC_FOLDER_NOT_OPEN
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: OnDemand returns a list of the display fields for the active folder. The list is a null-terminated character string with the tab character separating each field name.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
char data[1000], fields[10][100], * pToken;  
int j;  
  
if ( DoDdeCommand( "GET_DISPLAY_FIELDS", "", data ) )  
{  
    for ( pToken = strtok( data, "\t" ), j = 0;  
          pToken != NULL && j < 10;  
          pToken = strtok( NULL, "\t" ), j++ )  
    {  
        strcpy( fields[j], pToken );  
    }  
}
```

GET_DOC_VALUES

Command	Parameters
GET_DOC_VALUES	<i>/N doc number /S</i>

Parameters: **N**

Specifies the zero-based relative document number within the document list of the active folder. The number of documents in the list can be determined by using the GET_NUM_DOCS_IN_LIST command (refer to “GET_NUM_DOCS_IN_LIST” on page 222).

This parameter is required.

S

Indicates that values are to be returned only if the specified list item is selected.

This parameter is optional.

Action: None.

Return Code:

- 0** ARS_DDE_RC_NO_ERROR
- 2** ARS_DDE_RC_PARM_NOT_SPECIFIED
- 3** ARS_DDE_RC_INVALID_PARM
- 8** ARS_DDE_RC_FOLDER_NOT_OPEN
- 11** ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: OnDemand returns a list of the document values for the specified document within the document list of the active folder. The list is a null-terminated character string with the tab character separating each field value. The values are in the same sequence as the display field names returned by the GET_DISPLAY_FIELDS command (refer to “GET_DISPLAY_FIELDS” on page 217).

If the document number does not exist in the list or if the **S** parameter has been specified and the document number is not selected, no values are returned.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
char data[1000], values[10][100], * pToken;
int j;

if ( DoDdeCommand( "GET_DOC_VALUES", "8", data ) )
{
    for ( pToken = strtok( data, "\t" ), j = 0;
          pToken != NULL && j < 10;
          pToken = strtok( NULL, "\t" ), j++ )
    {
        strcpy( values[j], pToken );
    }
}
```

GET_FOLDER_FIELDS

Command	Parameters
GET_FOLDER_FIELDS	None.

Parameters: None.

Action: None.

Return Code:

- 0 ARS_DDE_RC_NO_ERROR
- 8 ARS_DDE_RC_FOLDER_NOT_OPEN
- 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: OnDemand returns a list of the folder fields for the active folder. The list is a null-terminated character string with the tab character separating each field name.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
char data[1000], fields[10][100], * pToken;
int j;

if ( DoDdeCommand( "GET_FOLDER_FIELDS", "", data ) )
{
    for ( pToken = strtok( data, "\t" ), j = 0;
          pToken != NULL && j < 10;
          pToken = strtok( NULL, "\t" ), j++ )
    {
        strcpy( fields[j], pToken );
    }
}
```

GET_FOLDERS

Command	Parameters
GET_FOLDERS	None.

Parameters: None.

Action: None.

Return Code:

0 ARS_DDE_RC_NO_ERROR
6 ARS_DDE_RC_NOT_LOGGED_ON
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: OnDemand returns a list of the available folders. The list is a null-terminated character string with the tab character separating each folder name.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
char data[1000], folders[10][31], * pToken;
int j;

if ( DoDdeCommand( "GET_FOLDERS", "", data ) )
{
    for ( pToken = strtok( data, "\t" ), j = 0;
          pToken != NULL && j < 10;
          pToken = strtok( NULL, "\t" ), j++ )
    {
        strcpy( folders[j], pToken );
    }
}
```

GET_NUM_DOCS_IN_LIST

Command	Parameters
GET_NUM_DOCS_IN_LIST	None.

Parameters: None.

Action: None.

Return Code:

- 0** ARS_DDE_RC_NO_ERROR
- 8** ARS_DDE_RC_FOLDER_NOT_OPEN
- 11** ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: OnDemand returns the number of documents currently in the document list of the active folder. The returned null-terminated string can be converted to a long integer.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
char data[100];
long num_docs;

if ( DoDdeCommand( "GET_NUM_DOCS_IN_LIST", "", data ) )
    num_docs = atol( data );
```

GET_NUM_DOC_PAGES

Command	Parameters
GET_NUM_DOC_PAGES	None.

Parameters: None.

Action: None.

Return Code:

0 ARS_DDE_RC_NO_ERROR
6 ARS_DDE_RC_NOT_LOGGED_ON
14 ARS_DDE_RC_NO_ACTIVE_DOC

Return Data: OnDemand returns the number of pages in the active document.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
char data[50];  
long num_pages;  
  
if ( DoDdeCommand( "GET_NUM_DOC_PAGES.", "", data ) )  
    num_pages = atol( data );
```

GET_PRINTERS

Command	Parameters
GET_PRINTERS	/L /S

Parameters: **L**

Indicates that a list of Local printers is to be returned. This parameter is optional, but either this or the **S** parameter must be specified.

S

Indicates that a list of Server printers is to be returned. This parameter is optional, but either this or the **L** parameter must be specified.

Action: None.

Return Code:

0 ARS_DDE_RC_NO_ERROR
2 ARS_DDE_RC_PARM_NOT_SPECIFIED
6 ARS_DDE_RC_NOT_LOGGED_ON
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: OnDemand returns a list of the available printers, either Local or Server printers as indicated by the parameter. If both parameters are specified, the results are unpredictable. The list is a null-terminated character string with the tab character separating each printer name.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
char data[1000], local_printers[10][100], * pToken;  
int j;  
  
if ( DoDdeCommand( "GET_PRINTERS", "/L", data ) )  
{  
    for ( pToken = strtok( data, "\t" ), j = 0;  
          pToken != NULL && j < 10;  
          pToken = strtok( NULL, "\t" ), j++ )  
    {  
        strcpy( local_printers[j], pToken );  
    }  
}
```

GET_QUERY_FIELDS

Command	Parameters
GET_QUERY_FIELDS	None.

Parameters: None.

Action: None.

Return Code:

0 ARS_DDE_RC_NO_ERROR
8 ARS_DDE_RC_FOLDER_NOT_OPEN
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: OnDemand returns a list of the query fields for the active folder. The list is a null-terminated character string with the tab character separating each field name.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
char data[1000], fields[10][100], * pToken;  
int j;  
  
if ( DoDdeCommand( "GET_QUERY_FIELDS", "", data ) )  
{  
    for ( pToken = strtok( data, "\t" ), j = 0;  
          pToken != NULL && j < 10;  
          pToken = strtok( NULL, "\t" ), j++ )  
    {  
        strcpy( fields[j], pToken );  
    }  
}
```

GET_SERVERS

Command	Parameters
GET_SERVERS	None.

Parameters: None.

Action: None.

Return Code:

- 0 ARS_DDE_RC_NO_ERROR
- 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: OnDemand returns a list of the available servers. The list is a null-terminated character string with the tab character separating each server name.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
char data[1000], servers[10][100], * pToken;
int j;

if ( DoDdeCommand( "GET_SERVERS", "", data ) )
{
    for ( pToken = strtok( data, "\t" ), j = 0;
          pToken != NULL && j < 10;
          pToken = strtok( NULL, "\t" ), j++ )
    {
        strcpy( servers[j], pToken );
    }
}
```


LOGOFF

Command	Parameters
LOGOFF	None.

Parameters: None.

Action: OnDemand performs a logoff.

Return Code:

- 0** ARS_DDE_RC_NO_ERROR
- 11** ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "LOGOFF", "", NULL );
```

LOGON

Command	Parameters
LOGON	<i>/S server name /U user id /P password /R</i>

Parameters: S

Specifies the logon server name. This parameter is required.

U

Specifies the logon user ID. This parameter is required.

P

Specifies the logon password. This parameter is required.

R

Indicates that if the logon fails, control is to be returned. This parameter is optional. If this parameter is not specified and the logon fails, the OnDemand Logon dialog box is displayed.

Action: OnDemand attempts a logon with the specified server name, user ID, and password. A list of the available server names can be retrieved by using the GET_SERVERS command (refer to “GET_SERVERS” on page 226).

If a user is already logged on, a logoff is performed prior to the logon. If the logon fails, the action is determined by the /R parameter.

Return Code:

- 0 ARS_DDE_RC_NO_ERROR
- 2 ARS_DDE_RC_PARM_NOT_SPECIFIED
- 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS
- 13 ARS_DDE_RC_USER_CANCELLED_OPERATION
- 25 ARS_DDE_RC_INVALID_USER_PASS_SERVER
- 26 ARS_DDE_RC_PASSWORD_EXPIRED

Return Data: If the logon is successful, OnDemand returns the user ID and password that were used to logon. The null-terminated character string contains the user ID, followed by a tab character, followed by the password.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
char parms[200];

sprintf( parms,
        "/S %s /U %s /P %s",
        "server26",
        "horatio",
        "vrxtwc" );

DoDdeCommand( "LOGON", parms, NULL );
```

OPEN_DOC

Command	Parameters
OPEN_DOC	<i>/N doc number /P path</i>

Parameters: N

Specifies the zero-based relative document number within the document list of the active folder. The number of documents in the list can be determined by using the GET_NUM_DOCS_IN_LIST command (refer to “GET_NUM_DOCS_IN_LIST” on page 222). The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command (refer to “GET_DOC_VALUES” on page 218).

This parameter is required.

P

Specifies the fully-qualified path of a file containing the document data.

If this parameter is specified, OnDemand does not access the database to retrieve a document, but it will retrieve a resource group if one is required.

This parameter is optional.

Action: OnDemand opens the document by displaying the first page in a document window.

Return Code:

- 0 ARS_DDE_RC_NO_ERROR
- 2 ARS_DDE_RC_PARM_NOT_SPECIFIED
- 3 ARS_DDE_RC_INVALID_PARM_VALUE
- 8 ARS_DDE_RC_FOLDER_NOT_OPEN
- 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS
- 12 ARS_DDE_RC_UNAUTHORIZED_OPERATION
- 13 ARS_DDE_RC_USER_CANCELLED_OPERATION

Return Data: If the document is successfully opened, OnDemand returns a *doc id*. This string contains a maximum of 20 characters.

This *doc id* is required as a parameter to other commands such as ACTIVATE_DOC and CLOSE_DOC.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
char doc_id[21];
```

```
DoDdeCommand( "OPEN_DOC", "/N 23", doc_id );
```

OPEN_FOLDER

Command	Parameters
OPEN_FOLDER	<i>/F folder name /S /C /R /D /V /P /T /A /N /L /0 button text /1 button text /2 button text /3 button text /4 button text /5 button text /6 button text /7 button text /8 button text /9 button text</i>

Parameters: F

Specifies the name of one of the folders available to the user on the server. This parameter is required.

S

Indicates that the Search button is **not** to appear on the Search Criteria section of the Search Criteria and Document List dialog box. This parameter is optional.

C

Indicates that the Clear All Fields button is **not** to appear on the Search Criteria section of the Search Criteria and Document List dialog box. This parameter is optional.

R

Indicates that the Restore Defaults button is **not** to appear on the Search Criteria section of the Search Criteria and Document List dialog box. This parameter is optional.

D

Indicates that the AND/OR buttons are **not** to appear on the Search Criteria section of the Search Criteria and Document List dialog box. This parameter is optional.

V

Indicates that the View All Selected button is **not** to appear on the Document List section of the Search Criteria and Document List dialog box. This parameter is optional.

P

Indicates that the Print All Selected button is **not** to appear on the Document List section of the Search Criteria and Document List dialog box. This parameter is optional.

T

Indicates that the Sort List ... button is **not** to appear on the Search Criteria section of the Search Criteria and Document List dialog box. This parameter is optional.

A

Indicates that the Audit buttons are **not** to appear on the Search Criteria section of the Search Criteria and Document List dialog box. This parameter is optional.

N

Indicates that the Append button is **not** to appear on the Document List section of the Search Criteria and Document List dialog box. This parameter is optional.

L

Indicates that the AutoScroll button is **not** to appear on the Document List section of the Search Criteria and Document List dialog box. This parameter is optional.

0 *button text*

Specifies the text for a button which is to appear on the Search Criteria section of the Search Criteria and Document List dialog box. This parameter is optional.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, OnDemand informs the client application through an Advise Loop established for the application. For more information on Advise Loops, see “DDEML Advise Loop” on page 258.

1 *button text*

Specifies the text for a button which is to appear on the Search Criteria section of the Search Criteria and Document List dialog box. This parameter is optional.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, OnDemand informs the client application through an Advise Loop established for the application. For more information on Advise Loops, see “DDEML Advise Loop” on page 258.

2 *button text*

Specifies the text for a button which is to appear on the Search Criteria section of the Search Criteria and Document List dialog box. This is an optional parameter.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, OnDemand informs the client application through an Advise Loop established for the application. For more information on Advise Loops, see “DDEML Advise Loop” on page 258.

3 *button text*

Specifies the text for a button which is to appear on the Search Criteria section of the Search Criteria and Document List dialog box. This is an optional parameter.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, OnDemand informs the client application through an Advise Loop established for the application. For more information on Advise Loops, see “DDEML Advise Loop” on page 258.

4 *button text*

Specifies the text for a button which is to appear on the Search Criteria section of the Search Criteria and Document List dialog box. This is an optional parameter.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, OnDemand informs the client application through an

Advise Loop established for the application. For more information on Advise Loops, “DDEML Advise Loop” on page 258.

5 *button text*

Specifies the text for a button which is to appear on the Document List section of the Search Criteria and Document List dialog box. This is an optional parameter.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, OnDemand informs the client application through an Advise Loop established for the application. For more information on Advise Loops, “DDEML Advise Loop” on page 258.

6 *button text*

Specifies the text for a button which is to appear on the Document List section of the Search Criteria and Document List dialog box. This is an optional parameter.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, OnDemand informs the client application through an Advise Loop established for the application. For more information on Advise Loops, see “DDEML Advise Loop” on page 258.

7 *button text*

Specifies the text for a button which is to appear on the Document List section of the Search Criteria and Document List dialog box. This is an optional parameter.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, OnDemand informs the client application through an Advise Loop established for the application. For more information on Advise Loops, see “DDEML Advise Loop” on page 258.

8 *button text*

Specifies the text for a button which is to appear on the Document List section of the Search Criteria and Document List dialog box. This is an optional parameter.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, OnDemand informs the client application through an Advise Loop established for the application. For more information on Advise Loops, see “DDEML Advise Loop” on page 258.

9 *button text*

Specifies the text for a button which is to appear on the Document List section of the Search Criteria and Document List dialog box. This is an optional parameter.

If the text includes an &, the following character becomes an accelerator key. When the user clicks on the button, OnDemand informs the client application through an Advise Loop established for the application. For more information on Advise Loops, see “DDEML Advise Loop” on page 258.

Action: OnDemand attempts to open the specified folder. A list of the available folder names can be retrieved by using the GET_FOLDERS command (refer to “GET_FOLDERS” on page 221). The Open a Folder dialog box is not displayed unless the open fails. If the folder is successfully opened, it becomes the active folder.

Multiple folders can be open concurrently. One of them is the active folder.

Return Code:

- 0 ARS_DDE_RC_NO_ERROR
- 2 ARS_DDE_RC_PARM_NOT_SPECIFIED
- 3 ARS_DDE_RC_INVALID_PARM_VALUE
- 6 ARS_DDE_RC_NOT_LOGGED_ON
- 7 ARS_DDE_RC_MAX_FOLDERS_OPEN
- 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "OPEN_FOLDER", "/F Mary's Folder", NULL );
```

PRINT_DOC

Command	Parameters
PRINT_DOC	<i>/N doc number /L printer name /S printer name /C copies /R orientation /K margins /M</i>

Parameters: **N** *doc number*

Specifies the zero-based relative document number within the document list of the active folder. The number of documents in the list can be determined by using the GET_NUM_DOCS_IN_LIST command (refer to “GET_NUM_DOCS_IN_LIST” on page 222). The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command (refer to “GET_DOC_VALUES” on page 218).

This parameter is optional. If it is not specified, the active document is printed.

L *printer name*

Specifies a Local printer name. The names of the available local printers can be determined by using the GET_PRINTERS command (refer to “GET_PRINTERS” on page 224).

This parameter is optional, but either this or the S parameter must be specified.

S *printer*

Specifies a Server printer name. The names of the available server printers can be determined by using the GET_PRINTERS command (refer to “GET_PRINTERS” on page 224).

This parameter is optional, but either this or the L parameter must be specified.

C *copies*

Specifies the number of copies of the document to be printed. The value must be a number between 1 and 100.

This parameter is optional. If not specified, one copy is printed.

R *orientation*

Specifies the document orientation for printing. For *orientation*, specify one of the following:

- B** rotates the document to best fit the paper.
- P** prints the document in the portrait orientation.
- L** prints the document in the landscape orientation.
- A** prints the document as specified in the printer.

If *orientation* is not specified, **B** is assumed.

This parameter is optional. If this parameter is not specified, **/R A** is assumed.

If a Server printer is specified, the **R** parameter is ignored.

K *margins*

Specifies the page margins to be used. For *margins*, specify ***t,b,l,r*** where:

- t*** is the top margin.
- b*** is the bottom margin.
- l*** is the left margin.
- r*** is the right margin.

Each margin value must be a non-negative decimal number. If no margin values are given, the current margin values are used.

This parameter is optional. If this parameter is not specified, margins of 0 (zero) are used. (A 0 margin may cause data truncation on some printers.)

If a Server printer is specified, the **K** parameter is ignored.

M

Indicates that the margins specified with the **K** parameter are given in millimeters rather than inches.

This parameter is optional. If the **M** parameter is not specified, the margins are assumed to be in inches.

Action: OnDemand prints the page(s) of the document on the specified printer.

Return Code:

- 0** ARS_DDE_RC_NO_ERROR
- 2** ARS_DDE_RC_PARM_NOT_SPECIFIED
- 3** ARS_DDE_RC_INVALID_PARM_VALUE
- 6** ARS_DDE_RC_NOT_LOGGED_ON
- 8** ARS_DDE_RC_FOLDER_NOT_OPEN
- 9** ARS_DDE_RC_NO_DOC
- 11** ARS_DDE_RC_USER_ACTION_IN_PROGRESS
- 12** ARS_DDE_RC_UNAUTHORIZED_OPERATION

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
char parms[200];

sprintf( parms,
        "/N %s /L %s /C %s /R B /K 0.5,1.2,1,1",
        "17",
        "HP LaserJet on LPT1.AES:",
        "2" );

DoDdeCommand( "PRINT_DOC", parms, NULL );
```

RESTORE_DEFAULTS

Command	Parameters
RESTORE_DEFAULTS	None.

Parameters: None.

Action: OnDemand sets the search criteria entry windows for the active folder to their default values.

Return Code:

0 ARS_DDE_RC_NO_ERROR
8 ARS_DDE_RC_FOLDER_NOT_OPEN
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "RESTORE_DEFAULTS", "", NULL );
```

RETRIEVE_DOC

Command	Parameters
RETRIEVE_DOC	<i>/N doc number /P doc path /R resgrp path /C combined path /A</i>

Parameters: N

Specifies the zero-based relative document number within the document list of the active folder. The number of documents in the list can be determined by using the GET_NUM_DOCS_IN_LIST command (refer to “GET_NUM_DOCS_IN_LIST” on page 222). The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command (refer to “GET_DOC_VALUES” on page 218).

This parameter is required.

P

Specifies the fully-qualified path of a file into which the document data is to be placed. This parameter is optional.

R

Specifies the fully-qualified path of a file into which the resource group data is to be placed. This parameter is optional.

C

Specifies the fully-qualified path of a file into which the combined resource group and data is to be placed. This parameter is optional.

A

Indicates that the data is to be appended to an existing file rather than replacing the file. This parameter is optional.

Action: OnDemand retrieves the resource group and/or document data and copies or appends it to the specified files. Any combination of the parameters may be specified, except that N is required.

Return Code:

- 0** ARS_DDE_RC_NO_ERROR
- 2** ARS_DDE_RC_PARM_NOT_SPECIFIED
- 3** ARS_DDE_RC_INVALID_PARM_VALUE
- 4** ARS_DDE_RC_SERVER_ERROR
- 5** ARS_DDE_RC_FILE_ERROR
- 8** ARS_DDE_RC_FOLDER_NOT_OPEN
- 9** ARS_DDE_RC_NO_DOC
- 11** ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
char parms[200];

sprintf( parms,
        "/N %d /C %s",
        26,
        "C:\\DATA\\COMBINED.FIL");

DoDdeCommand( "RETRIEVE_DOC", parms, NULL );
```


SEARCH_FOLDER

Command	Parameters
SEARCH_FOLDER	/A /O

Parameters: **A**

Indicates whether the documents resulting from the search are to be appended to the existing list. This parameter is optional. If not specified, the documents replace the previous list.

O

Indicates that the search criteria are to be ORed. This parameter is optional. If not specified, the search criteria are ANDed.

Action: OnDemand searches the active folder using the current search criteria.

Return Code:

0 ARS_DDE_RC_NO_ERROR
8 ARS_DDE_RC_FOLDER_NOT_OPEN
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "SEARCH_FOLDER", "", NULL );
```

SELECT_DOC

Command	Parameters
SELECT_DOC	<i>/N doc number</i>

Parameters: N

Specifies the zero-based relative document number within the document list of the active folder. The number of documents in the list can be determined by using the GET_NUM_DOCS_IN_LIST command (refer to “GET_NUM_DOCS_IN_LIST” on page 222). The values associated with a particular document number can be retrieved by using the GET_DOC_VALUES command (refer to “GET_DOC_VALUES” on page 218).

This parameter is required.

The *doc number* may be specified as -1 to indicate that all documents are to be selected.

Action: OnDemand selects (highlights) the Document List line that corresponds to the specific document number.

Return Code:

- 0 ARS_DDE_RC_NO_ERROR
- 2 ARS_DDE_RC_PARM_NOT_SPECIFIED
- 3 ARS_DDE_RC_INVALID_PARM
- 8 ARS_DDE_RC_FOLDER_NOT_OPEN
- 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "SELECT_DOC", "-1", NULL );
```

SET_FIELD_DATA

Command	Parameters
SET_FIELD_DATA	<i>/F field name /O operator /1 value1 /2 value2</i>

Parameters: F

Specifies the name of a search field in the active folder. This parameter is required.

A list of the field names can be retrieved by using the GET_QUERY_FIELDS command (refer to “GET_QUERY_FIELDS” on page 225).

O

Specifies the search operator to be used for the field. It must be one of the following:

EQ for Equal

NE for Not Equal

LT for Less Than

LE for Less Than or Equal

GT for Greater Than

GE for Greater Than or Equal

BW for Between

NB for Not Between

IN for In

NI for Not In

LK for Like

NL for Not Like

The operator must be one of those permitted for the field.

This parameter is optional. If not specified, the operator remains unchanged.

1

Specifies the value to be used for the first, and perhaps only, entry window for the field. This parameter is optional. If not specified, the value remains unchanged.

2

Specifies the value to be used for the second entry window for the field. This value is ignored if the search operator for the field is other than Between or Not Between. This parameter is optional. If not specified, the value remains unchanged.

Action: OnDemand updates the search operator, the first entry window, and/or the second entry window for the specified search field in the active folder.

Return Code:

0 ARS_DDE_RC_NO_ERROR
2 ARS_DDE_RC_PARM_NOT_SPECIFIED
3 ARS_DDE_RC_INVALID_PARM_VALUE
8 ARS_DDE_RC_FOLDER_NOT_OPEN
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
char parms[200];

sprintf( parms,
        "/F %s /O %s /1 %s /2 %s",
        "Account",
        "BW",
        "123456",
        "987654" );

DoDdeCommand( "SET_FIELD_DATA", parms, NULL );
```

SET_FOCUS

Command	Parameters
SET_FOCUS	None.

Parameters: None.

Action: OnDemand becomes the active window.

Return Code:

- 0 ARS_DDE_RC_NO_ERROR
- 11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "SET_FOCUS", "", NULL );
```

SET_HELP_PATH

Command	Parameters
SET_HELP_PATH	<i>/P path</i>

Parameters: P

Specifies the fully-qualified path for the Windows Help file.

This parameter is required.

Action: OnDemand invokes the specified Help file when the user requests help for one of the buttons or menu items associated with the client application.

The Help file is invoked with one of the following context IDs:

Search Criteria and Document List Dialog Box:

0x50800 ARS_DDE_HELP_ID_CRITERIA_BUTTON_1
0x50801 ARS_DDE_HELP_ID_CRITERIA_BUTTON_2
0x50802 ARS_DDE_HELP_ID_CRITERIA_BUTTON_3
0x50803 ARS_DDE_HELP_ID_CRITERIA_BUTTON_4
0x50804 ARS_DDE_HELP_ID_CRITERIA_BUTTON_5
0x50805 ARS_DDE_HELP_ID_DOCLIST_BUTTON_1
0x50806 ARS_DDE_HELP_ID_DOCLIST_BUTTON_2
0x50807 ARS_DDE_HELP_ID_DOCLIST_BUTTON_3
0x50808 ARS_DDE_HELP_ID_DOCLIST_BUTTON_4
0x50809 ARS_DDE_HELP_ID_DOCLIST_BUTTON_5

Toolbar and Menu Items:

0x5080A ARS_DDE_HELP_ID_SWITCH_FOCUS

Return Code:

0 ARS_DDE_RC_NO_ERROR
2 ARS_DDE_RC_PARM_NOT_SPECIFIED
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "SET_HELP_PARM", "C:\DDEAPPL\DDEAPPL.HLP", NULL);
```

SET_USER_MSG_MODE

Command	Parameters
SET_USER_MSG_MODE	/M <i>mode</i>

Parameters: M

Specifies the user message mode.

For *mode*, specify one of the following:

- 1 OnDemand always displays any user messages resulting from a DDE command.
- 2 OnDemand displays the user messages resulting from a DDE command only if the OnDemand window is visible and not minimized.
- 3 OnDemand suppresses any user messages resulting from a DDE command.

This parameter is required.

Action: OnDemand displays or suppresses subsequent user messages as specified by the parameters.

Return Code:

- 0 ARS_DDE_RC_NO_ERROR
- 2 ARS_DDE_RC_PARM_NOT_SPECIFIED
- 3 ARS_DDE_RC_INVALID_PARM_VALUE

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "SET_USER_MSG_MODE", "/M 2", NULL);
```

SHOW_WINDOW

Command	Parameters
SHOW_WINDOW	<i>/W placement</i>

Parameters: **W**

Specifies the show status and position of the OnDemand window. This parameter takes the same values as the Window Placement command line parameter. Refer to “Window Placement — /W placement” on page 187 for a description of these values.

If you specify this parameter without a value, the window is displayed at its most recent position with the most recent dimensions.

This parameter is required.

Action: OnDemand shows and/or positions its main window as specified by the parameter value.

Return Code:

0 ARS_DDE_RC_NO_ERROR
2 ARS_DDE_RC_PARM_NOT_SPECIFIED
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS

Return Data: None.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "SHOW_WINDOW", "/W 25,0,75,100", NULL );
```


STORE_DOC

Command	Parameters
STORE_DOC	<i>/P doc path /G appl group name /A appl name /n value /n value ...</i>

Parameters: P

Specifies the fully-qualified path of a file containing the document data to be stored in the OnDemand database. This parameter is required.

G

Specifies the name of an Application Group within the active folder. It is the responsibility of the caller to know the Application Group names associated with the active folder. This parameter is required.

A

Specifies the name of an Application within the specified Application Group. It is the responsibility of the caller to know the Application names associated with the specified Application Group. This parameter is required.

n

Specifies the value associated with a folder field. *n* is **x'01'** for the first field of the folder; **x'02'** for the second; and so forth. The associated *value* is a character string which can be converted to data of the field type (i.e., integer, date, etc.).

The number and order of folder fields can be determined by using the GET_FOLDER_FIELDS command. For more information on this command, see “GET_FOLDER_FIELDS” on page 220.

Any folder fields not specified are given an empty string for string fields or zero for numeric fields. If extraneous fields are specified, they are ignored.

Date fields must be provided in the format required for the field (for example, specifying 02/03/96 is invalid when February 3, 1996 is required).

If a slash character is to be included in a value, such as in a date, two consecutive slashes must be specified. For more information, see “Parameter Syntax” on page 185.

Action: OnDemand converts the folder field values to application group fields and stores the data from the specified file in the database as a document associated with the specified Application Group and Application.

Return Code:

- 0** ARS_DDE_RC_NO_ERROR
- 2** ARS_DDE_RC_PARM_NOT_SPECIFIED

```

4  ARS_DDE_RC_SERVER_ERROR
5  ARS_DDE_RC_FILE_ERROR
8  ARS_DDE_RC_FOLDER_NOT_OPEN
11 ARS_DDE_RC_USER_ACTION_IN_PROGRESS
12 ARS_DDE_RC_UNAUTHORIZED_OPERATION
15 ARS_DDE_RC_INVALID_APPL_GROUP_NAME
16 ARS_DDE_RC_INVALID_APPL_NAME
17 ARS_DDE_RC_INVALID_INTEGER_FIELD
18 ARS_DDE_RC_INVALID_DECIMAL_FIELD
19 ARS_DDE_RC_INVALID_DATE_FIELD
20 ARS_DDE_RC_INVALID_APPLGRP_FIELD_TYPE

```

Return Data: If the return code is one of the following:

```

ARS_DDE_RC_INVALID_INTEGER_FIELD
ARS_DDE_RC_INVALID_DECIMAL_FIELD
ARS_DDE_RC_INVALID_DATE_FIELD
ARS_DDE_RC_INVALID_APPLGRP_FIELD_TYPE

```

then OnDemand returns the relative folder field number of the invalid field. For example, if the first folder field is invalid, OnDemand returns a 0 (zero); if the second folder field is invalid, OnDemand returns a 1 (one); and so forth.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

C/C + + Example

```

char parms[200];

sprintf( parms,
        "/D %s /G %s /A %s /\x'01' %s /\x'02' %s /\x'03' %s",
        "D:\DATA\DOCDATA.AFP",
        "Student Data",
        "Grades",
        "Coed, Mary",
        "05//23//95",
        "3.15" );

DoDdeCommand( "STORE_DOC", parms, NULL );

```

Visual Basic Example

```

Dim cmdline As String
cmdline = "STORE_DOC /P D:\Data\DocData.AFP "
cmdline = cmdline + "/G Student Data "
cmdline = cmdline + "/A Grades "
cmdline = cmdline + "/" Chr(1) + " Coed.Mary "
cmdline = cmdline + "/" Chr(2) + " 05//23//95 "
cmdline = cmdline + "/" Chr(3) + " 3.15"

Call fncDDElink ( arstopic, cmdline, linktype, 3000 )

```

Note: For a definition of fncDDElink, see Appendix A, “Microsoft Visual Basic 5.0 DDE Program Sample” on page 323.

UPDATE_DOC

Command	Parameters
UPDATE_DOC	<i>/N doc number /F field name</i> <i>/V field value</i>

Parameters: N

Specifies zero-based relative document number within the document list of the active folder. This parameter is required.

The doc number may be specified as -1 to indicate that all selected documents are to be updated.

F

Specifies the name of a folder field. This parameter is required.

V

Specifies the value to be stored in the specified folder field. This value is will be converted to data of the field type (i.e., integer, date, etc.).

Date fields must be provided in the format required for the field (for example, specifying 02/03/96 is invalid when February 3, 1996 is required).

If a slash (/) character is to be included in a value (for example, in a date), two consecutive slashes must be specified. For more information, see “Parameter Syntax” on page 185.

This parameter is required.

Action: OnDemand converts the folder field value to an application group field and updates the data from the specified value in the database.

Return Code:

- 0 ARS_DDE_RC_NO_ERROR
- 3 ARS_DDE_RC_INVALID_PARM_VALUE
- 4 ARS_DDE_RC_SERVER_ERROR
- 8 ARS_DDE_RC_FOLDER_NOT_OPEN
- 12 ARS_DDE_RC_UNAUTHORIZED_OPERATION
- 20 ARS_DDE_RC_INVALID_APPLGRP_FIELD_TYPE

Return Data: OnDemand returns the number of documents that were successfully updated. The returned null-terminated string can be converted to a long integer.

Example: Refer to “DDEML Transactions” on page 195 for a description of the DoDdeCommand function.

```
DoDdeCommand( "UPDATE_DOC", "/N 1 /F Balance /V 123.45", NULL );
```

Return Codes

The following return codes are possible from DDE commands.

- 0** ARS_DDE_RC_NO_ERROR - Indicates that the command was executed without error.
- 1** ARS_DDE_RC_UNKNOWN_COMMAND - Indicates that the command was not a valid OnDemand DDE command.
- 2** ARS_DDE_RC_PARM_NOT_SPECIFIED - Indicates that a required parameter was not specified.
- 3** ARS_DDE_RC_INVALID_PARM_VALUE - Indicates that a parameter specified an invalid value.
- 4** ARS_DDE_RC_SERVER_ERROR - Indicates that a server error occurred while accessing the database.
- 5** ARS_DDE_RC_FILE_ERROR - Indicates that an error occurred during an input/output operation.
- 6** ARS_DDE_RC_NOT_LOGGED_ON - Indicates that a user must be logged on for the command to be executed.
- 7** ARS_DDE_RC_MAX_FOLDERS_OPEN - Indicates that the maximum number of folders are already open.
- 8** ARS_DDE_RC_FOLDER_NOT_OPEN - Indicates that a folder must be open and active for the command to be executed.
- 9** ARS_DDE_RC_NO_DOC - Indicates that there is no database data associated with a document in the document list.
- 11** ARS_DDE_RC_USER_ACTION_IN_PROGRESS - Indicates that OnDemand is busy performing a user-initiated action. A modal dialog box or message box is probably being displayed.
- 12** ARS_DDE_RC_UNAUTHORIZED_OPERATION - Indicates that the user currently logged on is not authorized to perform the requested operation.
- 13** ARS_DDE_RC_USER_CANCELLED_OPERATION - Indicates that the user cancelled an operation requested through DDE.
- 14** ARS_DDE_RC_NO_ACTIVE_DOC - Indicates that there is no currently active document.
- 15** ARS_DDE_RC_INVALID_APPL_GROUP_NAME - Indicates that the application group name provided is not valid for the folder.
- 16** ARS_DDE_RC_INVALID_APPL_NAME - Indicates that the application name provided is not valid for the application group.
- 17** ARS_DDE_RC_INVALID_INTEGER_FIELD - Indicates that the value provided for a folder field is not a valid integer.
- 18** ARS_DDE_RC_INVALID_DECIMAL_FIELD - Indicates that the value provided for a folder field is not a valid decimal.

- 19** ARS_DDE_RC_INVALID_DATE_FIELD - Indicates that the value provided for a folder field is not a valid date.
- 20** ARS_DDE_RC_INVALID_APPLGRP_FIELD_TYPE - Indicates that the field type within the application group is not valid.
- 21** ARS_DDE_RC_DOC_NOT_VIEWABLE_OR_PRINTABLE - Indicates that the specified document cannot be displayed or printed using the DDE interface.
- 22** ARS_DDE_RC_INCORRECT_CURRENT_PASSWORD - Indicates that the current password specified for the user is incorrect.
- 23** ARS_DDE_RC_PASSWORD_TOO_SHORT - Indicates that the new password specified is not long enough.
- 24** ARS_DDE_RC_NEW_PASSWORD_MISMATCH - Indicates that the two passwords specified for the new password do not match.
- 25** ARS_DDE_RC_INVALID_USER_PASS_SERVER - Indicates that the userid, password, or server was not valid.
- 26** ARS_DDE_RC_PASSWORD_EXPIRED - Indicates that the password has expired.

DDEML Advise Loop

The client application may create a DDEML Advise Loop in order to be informed when one of the following events occur:

Code	Event
------	-------

S	User has switched focus to the client application using the first menu item/toolbar button.
S2	User has switched focus to the client application using the second menu item/toolbar button.
S3	User has switched focus to the client application using the third menu item/toolbar button.
S4	User has switched focus to the client application using the fourth menu item/toolbar button.
S5	User has switched focus to the client application using the fifth menu item/toolbar button.
0	User has clicked Search Criteria button 1 on the Search Criteria and Document List dialog box.
1	User has clicked Search Criteria button 2 on the Search Criteria and Document List dialog box.
2	User has clicked Search Criteria button 3 on the Search Criteria and Document List dialog box.
3	User has clicked Search Criteria button 4 on the Search Criteria and Document List dialog box.
4	User has clicked Search Criteria button 5 on the Search Criteria and Document List dialog box.
5	User has clicked Document List button 5 on the Search Criteria and Document List dialog box.
6	User has clicked Document List button 1 on the Search Criteria and Document List dialog box.
7	User has clicked Document List button 2 on the Search Criteria and Document List dialog box.
8	User has clicked Document List button 3 on the Search Criteria and Document List dialog box.
9	User has clicked Document List button 4 on the Search Criteria and Document List dialog box.

The DDEML DdeClientTransaction function is used, with ADV_START and ADV_STOP transactions, to start and stop an Advise Loop. The DDEML item name string must contain "1". The client application may initiate an Advise Loop after establishing connection to OnDemand and maintain it for the duration of the conversation or start and stop the loop at its discretion. Notification of events takes place only when a loop is active.

The client application receives notification of an event through an XTYP_ADVDATA transaction at its DDEML callback function. The data returned is a null-terminated character string containing the **Code** for the **Event**.

External Applications and Dynamic Link Libraries

For Windows 95/NT, OnDemand provides menu and toolbar extensions that allow an end user to invoke another Windows application or execute a function in a Dynamic Link Library (DLL). This facility is activated by placing information in the Windows system registry. If during initialization, OnDemand detects this information, it adds menu items and toolbar buttons. When the user chooses one of these menu items or clicks one of these toolbar buttons, OnDemand invokes the associated application or calls the associated entry point in a DLL.

OnDemand looks in the Windows system registry for the following keys and string values:

Key	Value Name	Value Data
HKEY_CURRENT_USER		
or		
HKEY_LOCAL_MACHINE		
Software		
IBM		
OnDemand32		
Client		
	ExternalApps Apps	<i>a1[,a2][,a3][,a4][,a5]</i>
<i>a1</i>	Path	<i>application path</i>
	MenuText	<i>menu item text</i>
	BitmapDLL	<i>toolbar button bitmap DLL path</i>
	BitmapResid	<i>toolbar button bitmap res id</i>
	Folders	<i>folder name[\\folder name]...</i>
	ExcludeFolders	<i>folder name[\\folder name]...</i>
	Doc	0 1 2 3
	CopyDoc	ASIS ASCII
	Parameter	<i>parameter data</i>
.		
.		
	ExternalDlls Dlls	<i>d1[,d2][,d3][,d4][,d5]</i>
<i>d1</i>	Path	<i>DLL path</i>
	Function	<i>function name</i>
	MenuText	<i>menu item text</i>
	BitmapDLL	<i>toolbar button bitmap DLL path</i>
	BitmapResid	<i>toolbar button bitmap res id</i>
	Folders	<i>folder name[\\folder name]...</i>
	ExcludeFolders	<i>folder name[\\folder name]...</i>
	Doc	0 1 2 3
	CopyDoc	ASIS ASCII
	Parameter	<i>parameter data</i>
.		
.		

A maximum of five applications and five DLLs may be specified. They may be divided between the **HKEY_CURRENT_USER** and **HKEY_LOCAL_MACHINE** keys. The key names *a1*, ..., *a5* and *d1*, ..., *d5* may be any string. The values, which must be string values, are interpreted as follows:

- **Path** specifies the path for the application or DLL. This value must be provided and should be fully-qualified or in the execution path.
- **Function** specifies the name of an entry point in the DLL specified with **Path**. This value must be provided for a DLL. It is not relevant for an application.
- **MenuText** specifies the text to appear on the associated menu item under the OnDemand Window menu. This value is optional. If not provided, the menu item is blank.
- **BitmapDLL** specifies the path for a DLL containing a bitmap resource to be used for a toolbar button to be associated with the application or DLL. This DLL may be the same as or different from any DLL specified for a **Path** DLL or another **BitmapDLL** DLL. The bitmap should be 16 pels wide and 16 pels high.

This value is optional. If not provided, a toolbar button will not be created.

- **BitmapResid** specifies the resource id of the bitmap within the DLL specified for **BitmapDLL**. This value is ignored if **BitmapDLL** is not specified and optional if it is. If not provided, a value of 0 is assumed.
- **Folders** specifies one or more names of OnDemand folders. If multiple names are provided, they must be separated by a backslash ('\') character. An asterisk ('*') may be used as a wildcard character in the last position of the name. This is equivalent to listing all folder names beginning with the characters preceding the asterisk.

The associated menu item, and corresponding toolbar button, is enabled whenever: 1) a document is being viewed and the folder associated with the active document is one of the specified folders, or 2) a document is not being viewed and the current folder is one of the specified folders.

This value is optional. If the **ExcludeFolders** value is provided, this value is ignored. If neither is provided, a folder name test is not performed before enabling the menu item and toolbar button.

- **ExcludeFolders** specifies one or more names of OnDemand folders. The syntax is the same as the **Folders** value.

The associated menu item, and corresponding toolbar button, is enabled whenever: 1) a document is being viewed and the folder associated with the active document is **not** one of the specified folders, or 2) a document is not being viewed and the current folder is **not** one of the specified folders.

This value is optional. If not provided, enablement is controlled by the **Folders** value.

- **Doc** may be specified as one of the following values:
 - **0** indicates that enablement of the associated menu item and corresponding toolbar button is limited only by the **Folders** and **ExcludeFolders** values.

1 indicates that the associated menu item and corresponding toolbar button is enabled only when a document is being viewed.

2 indicates that the associated menu item and corresponding toolbar button is enabled only when a document list is being viewed and at least one document is selected.

3 indicates that the associated menu item and corresponding toolbar button is enabled only when a document is being viewed or a document list is being viewed and at least one document is selected.

This value is optional. If not provided, a value of **3** is assumed.

- **CopyDoc** indicates that a copy of one or more documents is to be provided to the external application or DLL and specifies the type of data to be provided. If the value is **ASIS**, the document data is in its native format. If the value is **ASCII**, the document data is converted to an ASCII file.

If the end user chooses the associated menu item or corresponding toolbar button when a document is being viewed, the data provided is for the active document. If the folder document list is being displayed, the data is a concatenation of all documents selected in the document list.

This value is optional. If not provided, no document is provided to the external application or DLL.

- **Parameter** specifies a maximum of 255 characters to be passed as parameter data to the external application or DLL. This value is optional. If not provided, no parameter data is passed to the external application or DLL.

The following is an example of the required registry entries:

Key	Value Name	Value Data
HKEY_CURRENT_USER		
Software		
IBM		
OnDemand32		
Client		
ExternalApps	Apps	KensApp,Edith
KensApp	Path	C:\KENS DIR\KEN
	MenuText	Ken's Application
	BitmapDLL	C:\RES DLLS\RES DLL1.DLL
	BitmapResid	27
	Doc	1
Edith	Path	E:\EDITH\EDITH.EXE
	MenuText	Edith Spreadsheet Stuff
	Folders	Edith Folder
ExternalDlls	Dlls	MyFunc
MyFunc	Path	D:\MYFUNC\MYFUNC.DLL
	Function	MyFuncEntryPoint
	MenuText	My Function
	BitmapDLL	D:\MYFUNC\MYFUNC.DLL
	BitmapResid	14
	CopyDoc	ascii
	Parameter	Parameter Data

If the user chooses a menu item or clicks a toolbar button associated with an application, OnDemand invokes the application with the command line:

application path /F folder /P page /T text /A attrvalue /D fields /N filename /R data

Where:

- *application path* is the path specified with the **Path** value in the registry.
- *folder* is the name of the active folder or, if a document is being displayed, the folder associated with that document. If no folder is open, the **/F** parameter is not provided.
- *page* is the current page of the document being viewed. If no document is being viewed or if the active folder window is being displayed, the **/P** parameter is not provided.
- *text* is the currently selected text in the document being viewed or the document list of the active folder. The format of the text is the same as that obtained by copying it to the clipboard. It may contain tab and newline characters. If no such text exists, the **/T** parameter is not provided.
- *attrvalue* is the attribute/value pair associated with the current page of the document being viewed. The attribute and value are separated by a tab character. If no such attribute/value pair exists, the **/A** parameter is not provided.
- *fields* is a set of folder field name/value pairs associated with the document being viewed. Each pair contains the name of a folder field and, separated by a tab character, the value of that field for the document. The pairs are separated by a newline character. If no such text exists, the **/T** parameter is not provided.
- *filename* is the fully-qualified path and filename of document data as described for the **CopyDoc** registry entry. If the **CopyDoc** entry is not specified or no data is available, the **/N** parameter is not provided.
- *data* is the parameter data specified for the **Parameter** registry entry. If the **Parameter** entry is not specified, the **/R** parameter is not provided.

If the user chooses a menu item or clicks a toolbar button associated with a DLL, OnDemand calls the entry point (function) specified with the **Function** value in the registry. This function must be of the following type:


```
typedef void ( WINAPI * ArsExternalDllFunction )
    ( long  page_number,
      char * pFolderName,
      char * pSelectedText,
      char * pAttributeAndValue,
      char * pFieldsAndValues,
      char * pFilename,
      char * pParameterData );
```

Where:

- *page_number* is the current page of the document being viewed. If no document is being viewed or if the active folder window is being displayed, this value is 0.
- *pFolderName* is a pointer to a null-terminated character string containing the name of the active folder or, if a document is being displayed, the folder associated with that document. If no folder is open, this value is NULL.
- *pSelectedText* is a pointer to a null-terminated character string containing the currently selected text in the document being viewed or the document list of the active folder. The format of the text is the same as that obtained by copying it to the clipboard. It may contain tab and newline characters. If no such text exists, this value is NULL.
- *pAttributeAndValue* is a pointer to a null-terminated character string containing the attribute/value pair associated with the current page of the document being viewed. The attribute and value are separated by a tab character. If no such attribute/value pair exists, this value is NULL.
- *pFieldsAndValues* is a pointer to a null-terminated character string containing a set of folder field name/value pairs associated with the document being viewed. Each pair contains the name of a folder field and, separated by a tab character, the value of that field for the document. The pairs are separated by a newline character. If no such text exists, this value is NULL.
- *pFilename* is a pointer to a null-terminated character string containing the fully-qualified path and filename of document data as described for the **CopyDoc** registry entry. If the **CopyDoc** entry is not specified or no data is available, this value is NULL.
- *pParameterData* is a pointer to a null-terminated character string containing the parameter data specified for the **Parameter** registry entry. If the **Parameter** entry is not specified, this value may be NULL or point to an empty string.

Related Documents

For Windows 95/NT, OnDemand provides menu and toolbar extensions that allow an end user to retrieve and view a document related to the document currently being viewed. This facility is activated by placing information in the Windows system registry. If during initialization, OnDemand detects this information, it adds menu items and toolbar buttons. When the user chooses one of these menu items or clicks one of these toolbar buttons, OnDemand retrieves the related document and displays it along with the original document.

A typical application for Related Documents is a credit card folder containing monthly statements. The end user views a customer's statement which has a line for each credit card transaction, selects a transaction number on one of these lines, and clicks a toolbar button. OnDemand searches an associated transaction folder for the customer's account number and selected transaction number, then displays the transaction document alongside the statement.

OnDemand looks in the Windows system registry for the following keys and string values:

Key	Value Name	Value Data
HKEY_CURRENT_USER		
or		
HKEY_LOCAL_MACHINE		
Software		
IBM		
OnDemand32		
Client		
	RelatedDocs Related	<i>r1[,r2][,r3][,r4][,r5][,r6][,r7][,r8][,r9][,r10]</i>
<i>r1</i>	MenuText	<i>menu item text</i>
	BitmapDLL	<i>toolbar button bitmap DLL path</i>
	BitmapResid	<i>toolbar button bitmap res id</i>
	Folders	<i>folder name[folder name]...</i>
	ExcludeFolders	<i>folder name[folder name]...</i>
	RelatedFolder	<i>related folder name</i>
	Fields	<i>field information</i>
	Arrange	V H M C U
	.	
	.	
	.	

A maximum of ten Related Documents may be specified. They may be divided between the HKEY_CURRENT_USER and HKEY_LOCAL_MACHINE keys. The key names *r1*, ..., *r10* may be any string. The values, which must be string values, are interpreted as follows:

- **MenuText** specifies the text to appear on the associated menu item under the OnDemand Window menu. This value is optional. If not provided, the menu item is blank.
- **BitmapDLL** specifies the path for a DLL containing a bitmap resource to be used for a toolbar button to be associated with the Related Documents. The bitmap should be 16 pels wide and 16 pels high.

This value is optional. If not provided, a toolbar button will not be created.

- **BitmapResid** specifies the resource id of the bitmap within the DLL specified for **BitmapDLL**. This value is ignored if **BitmapDLL** is not specified and optional if it is. If not provided, a value of 0 is assumed.
- **Folders** specifies one or more names of OnDemand folders. If multiple names are provided, they must be separated by a backslash (`\`) character. An asterisk (`*`) may be used as a wildcard character in the last position of the name. This is equivalent to listing all folder names beginning with the characters preceding the asterisk.

The associated menu item, and corresponding toolbar button, is enabled whenever: 1) a document is being viewed and the folder associated with the active document is one of the specified folders, or 2) a document is not being viewed and the current folder is one of the specified folders.

This value is optional. If the **ExcludeFolders** value is provided, this value is ignored. If neither is provided, a folder name test is not performed before enabling the menu item and toolbar button.

- **ExcludeFolders** specifies one or more names of OnDemand folders. The syntax is the same as the **Folders** value.

The associated menu item, and corresponding toolbar button, is enabled whenever: 1) a document is being viewed and the folder associated with the active document is **not** one of the specified folders, or 2) a document is not being viewed and the current folder is **not** one of the specified folders.

This value is optional. If not provided, enablement is controlled by the **Folders** value.

- **RelatedFolder** specifies the name of an OnDemand folder which contains a document related to the document being viewed. This folder may be the same as or different from the folder specified by **Folders**.
- **Fields** specifies information describing the document to be retrieved from the related folder. This information is used to perform a search of the folder. The folder fields are first initialized to their default operator and values. The fields specified here are then set

to the operator and values requested. The first document in the document list resulting from the search is considered to be the related document.

The information has the following format:

fieldname=operator\value1[\value2]; ... ;fieldname=operator\value1[\value2]

Where:

- *fieldname* is the name of a field defined for the related folder.
 - *operator* is the search operator to be used for the field. It must be one of the following values:
 - EQ** for Equal
 - NE** for Not Equal
 - GT** for Greater Than
 - GE** for Greater Than Or Equal
 - LT** for Less Than
 - LE** for Less Than Or Equal
 - BW** for Between
 - NB** for Not Between
 - IN** for In
 - NI** for Not In
 - LK** for Like
 - NL** for Not Like
 - *value1* is the first or only value to be used for the field. It may be any string, including an empty string, and may contain the following substitution characters.
 - %v** If the original folder contains a field of the same name, the value of that field for the original document is substituted in place of these characters; otherwise, an empty string is substituted.
 - %s** These characters are replaced by any text which the user has currently selected in the original document.
 - %%** These characters are replaced by a single %.
 - *value2* is the second value to be used for the field. It is ignored unless the operator is Between or Not Between. The format is the same as *value1*.
- **Arrange** specifies the arrangement of the document windows after the related document is brought into view. The value must be one of the following:
 - V** for tiled vertically.
 - H** for tiled horizontally.
 - M** for maximized with the related document overlaying the original.
 - C** for cascaded.

U for left to the user's specification via the **Maintain Document Arrangement** menu item.

The following is an example of the required registry entries:

Key	Value Name	Value Data
HKEY_CURRENT_USER		
Software		
IBM		
OnDemand32		
Client		
	RelatedDocs	Related Credit,Students
	Credit	MenuText Transaction Detail BitmapDLL C:\RESDLLS\RESDLL1.DLL BitmapResid 27 Folders Credit Card Statements RelatedFolder Credit Card Transactions Fields Account=lk\%v;Transaction=eq\%s Arrange v
	Students	MenuText Grades BitmapDLL C:\RESDLLS\RESDLL1.DLL BitmapResid 56 Folders Student Information RelatedFolder Student Information Fields ID=eq\%v;Document Type=eq\GRADES Arrange h

Product Information File

A Product Information File (PIF) can be created to customize the OnDemand application title and the appearance of the “About” dialog box that appears during initialization and when the user selects the About item in the Help menu.

A PIF has the name `PRODUCT.INF` and is found in the same directory from which OnDemand is executed. The PIF has the same format and syntax as a standard Windows INI file. It contains a single section named `PRODUCT`. The section contains the following entries:

- `NAME` specifies the title to be used at the top of the main OnDemand window and on a number of menu items.
- `LOGO_FILE` specifies a fully-qualified path of a Device Independent Bitmap (DIB) file to be used as the logo in the About dialog box.
- `ABOUT_TITLE` specifies a title to be used for the About dialog box.
- `ABOUT_LINE n` specifies each line of the About dialog box text. `ABOUT_LINE1` gives the first line; `ABOUT_LINE2` gives the second; and so forth through `ABOUT_LINE8`.

Document Audit Facility

Overview

The Document Audit Facility (DAF) can be used to audit documents stored in OnDemand. To use the DAF, you must first create a control file and define the reports to be audited to OnDemand. After you load the reports into the system, you can use the Windows client to audit the documents. When you retrieve a document from a folder defined in the DAF control file, OnDemand displays two additional command buttons on the client viewing window. One button is used to mark documents that pass the audit. The other button is used to mark documents that fail the audit.

Note: Users that need to audit documents must be given permission to update documents. See “Controlling access to the DAF” on page 278 for more information.

The following topics contain additional information:

- Creating the DAF control file
- Defining the report
- Controlling access to the DAF
- Using the DAF

Creating the DAF control file

The DAF is controlled by a file named ARSGUI.CFG, which you must create and store in the Windows client program directory (\Program Files\IBM\OnDemand32 by default). The DAF file has the same format and syntax as a standard Windows INI file. The DAF file contains a section named AUDIT, which identifies one or more folder sections. Each folder section identifies a folder that can be audited. Figure 1 on page 276 shows a sample DAF file.

```

[AUDIT]
FOLDERS=LDR,Student Information

[LDR]
FOLDER=Loan Delinquency Report
AUDIT_FIELD=Document Audit
PASS_TEXT=Pass
FAIL_TEXT=Fail
PASS_VALUE=P
FAIL_VALUE=F

[Student Information]
FOLDER=Student Information
AUDIT_FIELD=Audit Status
PASS_VALUE=P
FAIL_VALUE=F

```

Figure 1. Sample DAF (ARSGUI.CFG) File

The AUDIT section

The AUDIT section contains one record, the FOLDERS record. The FOLDERS record contains a comma-separated list of folder section names. You must create an additional section in the DAF file for each folder section named in the FOLDERS record. The total number of characters in the FOLDERS record must not exceed 255.

The folder section

Each folder section contains the following records:

- FOLDER specifies the name of the folder, exactly as it appears in OnDemand. The FOLDER record is required.
- AUDIT_FIELD specifies the name of the folder field used to audit documents, exactly as it appears in OnDemand. See “Defining the Folder” on page 278 for more information. The AUDIT_FIELD record is required.
- PASS_TEXT is the caption that appears on the command button used to mark a document that passes an audit. The total number of characters in the PASS_TEXT record must not exceed 50. The PASS_TEXT record is optional. The default caption is Pass.
- FAIL_TEXT is the caption that appears on the command button used to mark a document that fails an audit. The total number of characters in the FAIL_TEXT record must not exceed 50. The FAIL_TEXT record is optional. The default caption is Fail.
- PASS_VALUE is the value that is stored in the database for documents that pass an audit. The value is stored in the application group field. See “Defining the application group”

on page 277 for more information. The total number of characters in the PASS_VALUE record must not exceed 254. The PASS_VALUE record is required.

- FAIL_VALUE is the value that is stored in the database for documents that fail an audit. The value is stored in the application group field. See “Defining the application group” for more information. The total number of characters in the FAIL_VALUE record must not exceed 254. The FAIL_VALUE record is required.

Defining the report

The following topics provide information that you need to specify when you define a report that uses the DAF. The information provided is in addition to all of the other attributes you need to specify when you define a report to OnDemand.

- Defining the application group
- Defining the application
- Defining the folder

Defining the application group

Add the audit field to the application group on the Field Definition page. The field type must be STRING.

Define the attributes of the audit field on the Field Information page.

- In the String area, set the Case to Upper, Type to Fixed, and Length to 1 (one).
- In the Mapping area, add the Database and Displayed Values for the audit field. Database values are stored in the database and Displayed Values appear in search fields and the document list. Enter a Database Value and its corresponding Displayed Value in the spaces provided. Click Add to add each pair of values to the application group.

You must add one set of values for documents that pass an audit (PASS), one set of values for documents that fail an audit (FAIL), and one set of default values:

- The PASS Database Value must match the value of the PASS_VALUE record in the folder section of the DAF file. We recommend that the PASS Displayed Value match the value of the PASS_TEXT record in the folder section of the DAF file.
- The FAIL Database Value must match the value of the FAIL_VALUE record in the folder section of the DAF file. We recommend that the FAIL Displayed Value match the value of the FAIL_TEXT record in the folder section of the DAF file.
- The default values are used to set the status of all documents when a report is loaded into the application group. You typically set the default values to N, for None or Not Audited.

Table 1 on page 278 shows an example.

Table 1. Database and Displayed Values

Database Value	DAF File	Displayed Value	DAF File
F	FAIL_VALUE=F	Fail	FAIL_TEXT=Fail
P	PASS_VALUE=P	Pass	PASS_TEXT=Pass
N		Not Audited	

Note: The N Database Value and the Not Audited Displayed Value are not stored in the DAF file.

Defining the application

Add the default value for the audit field on the Load Information page. This is the value that OnDemand stores in the database for all documents when a report is loaded into the application group. We recommend that you set the default value to N (for None or Not Audited).

The default value must match the default Database Value for the audit field. You defined the default Database Value for the audit field on the application group Field Information page. See “Defining the application group” on page 277 for more information.

Defining the Folder

Add the audit field to the folder on the Field Definition page. The field type must be STRING. The name of the field must match the value of the AUDIT_FIELD record in the folder section of the DAF file. See “The folder section” on page 276 for more information.

Map the folder audit field to the application group audit field on the Field Mapping page.

Controlling access to the DAF

There are several ways you can manage access to documents that need to be audited. However, to simplify administration of the system, we strongly encourage you to use groups. For example, some customers will define two groups:

- **Viewers.** At a minimum, users in the Viewers group are not permitted to audit documents. In addition, some customers create a default logical view so that the audit field does not appear when users in the Viewers group open a document. Other customers define a query restriction so that users in the Viewers group see only documents that have passed an audit.
- **Auditors.** The users in the Auditors group are permitted to audit documents. Users in the Auditors group must be given permission to update documents in the application groups that contain documents to be audited.

If the requirements of your system are not met by these two groups, you may need to define additional groups or configure the system differently. Contact the IBM support center if you have questions about users, groups, or other aspects of administering the system.

Using the DAF

After a report is loaded into OnDemand, authorized users can use the DAF to audit the documents. To audit a document, open one of the folders defined in the DAF control file. Search the folder for documents that need to be audited. For example, search for documents that contain the value `Not Audited` in the audit field. Select and view one or more documents from the document list. In the document viewing window, click the `Pass` button to mark a document that passes the audit; click the `Fail` button to mark a document that fails the audit. OnDemand updates the database with the pass/fail value specified in the DAF control file.

Integration with Monarch Version 5

This section provides information about how to integrate Monarch Version 5 with the OnDemand Windows client. This function allows users to automatically load documents from the OnDemand client into Monarch. The user can then do complex data manipulation from Monarch, such as creating derived columns and generating charts and reports.

Note: Monarch is a software program that is available from Datawatch Corporation.

This section is of primary interest to administrators responsible for installing, configuring, and distributing software products. This section shows the steps that you typically need to take to integrate Monarch with OnDemand. This section describes how to do some of the tasks, but you will need other OnDemand information and your Monarch information to do others.

An administrator can use the OnDemand Windows 32-bit client installation program to distribute Monarch files with the OnDemand client and configure PCs to run Monarch from the OnDemand client. We recommend that an administrator configure a copy of the OnDemand client software on a reference workstation ¹ and configure the installation program on a distribution server.² Each user planning to run Monarch from OnDemand must have the Monarch software installed on their PC before they install or upgrade the OnDemand client with this function.

An administrator can store the Monarch files in the OnDemand Windows client installation directory tree on a distribution server. The installation program copies the Monarch files along with the standard OnDemand client files when the user installs the client from the server to the PC. You can distribute several types of files, including:

- Registry files. The installation program imports these files into the Registry on the user's PC. A Registry file named ODMonarch.Reg that contains information about the Monarch DLL for the OnDemand client must be present for the installation program to integrate OnDemand and Monarch on the user's PC.

Important: It is easy to accidentally destroy important data and render a system completely unusable by importing Registry files. Plan to backup the Registry on the user's PC before they run the Setup program.

- Monarch model files. The installation program copies these files to the Monarch\Models directory on the user's PC.

¹ A reference workstation contains an installed copy of the OnDemand client with the Monarch DLL defined to OnDemand. An administrator uses configuration information from the reference workstation to distribute software to other users.

² A distribution server is a network file server that contains a copy of the OnDemand client installation software in a shared location. Other users on the network use the copy to run the OnDemand installation program and install the client from the distribution server to their PCs.

The following topics provide additional information:

- Before you begin
- Configuring the OnDemand client software
- Configuring the OnDemand Setup program
- Running the OnDemand Setup program
- Running Monarch from OnDemand
- Upgrading the client

Before you begin

Important: If your organization has integrated Monarch with OnDemand Version 2.2.1.2 or earlier, please see “Upgrading your OnDemand client” on page 291.

Before you continue, you should have already completed the following tasks:

- Installed Monarch Version 5 on the reference workstation. See your Monarch information for details.
- Obtained the latest PTF for the OnDemand Windows client from IBM service on the World Wide Web. To do so, point your Web browser to:

`ftp://www.service.software.ibm.com/software/ondemand/fixes`

Then follow the links to the latest PTF for the client. Click on the `odwin32.zip` file and save it to disk.

Configuring the client

This chapter describes how to configure the client on the reference workstation with the information needed to integrate Monarch with OnDemand. If you have not done so already, install a copy of Monarch Version 5 (see your Monarch information for details). Then install a copy of the OnDemand client software (see your other OnDemand information for details). Next, add the Registry key, values, and value data that define the properties of the Monarch DLL to the OnDemand client. Then verify that you have correctly integrated Monarch with OnDemand by starting the OnDemand client, opening a document, and invoking Monarch using the associated menu command or toolbar button. When you are satisfied that everything is working correctly, export the Registry key to a file. The following topics provide more information:

- Adding the Registry key
- Exporting the Registry key
- Using multiple Monarch model files

Adding the Registry key

Note: You must install the OnDemand Windows client on the reference workstation before you add the Registry key.

Placing information in the Registry activates Monarch integration with the OnDemand client. If during initialization, OnDemand detects this information, it adds menu items and toolbar buttons to the client workspace. When the user chooses one of the menu items or clicks one of the toolbar buttons, OnDemand calls the associated entry point in the Monarch DLL. OnDemand looks in the Registry under the **ExternalDlls** subkey to determine the action to take and other information. Figure 2 shows an example of the ExternalDlls subkey.

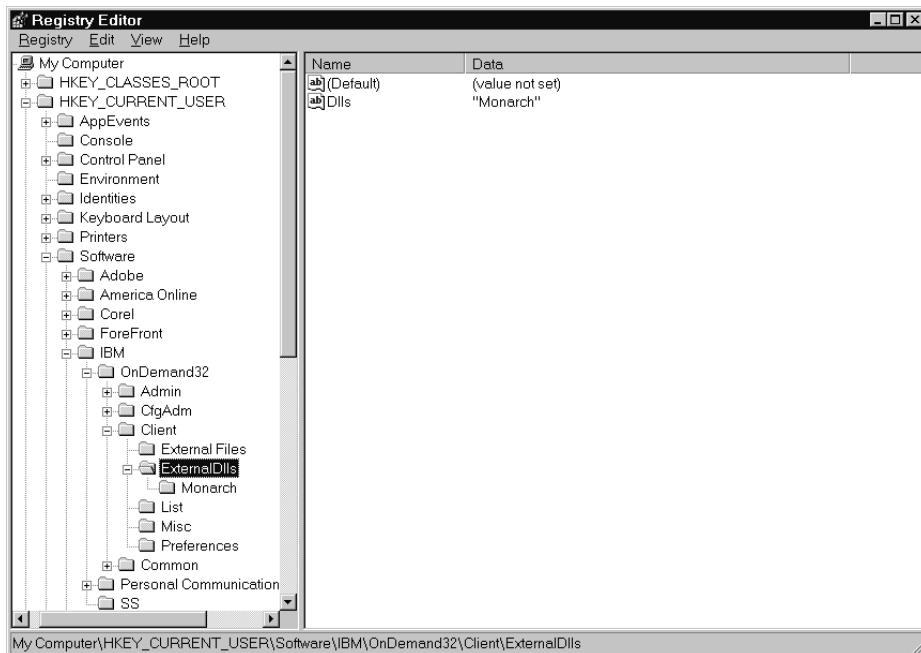


Figure 2. ExternalDlls subkey showing one value in Dlls

Subkeys under ExternalDlls contain the information used by OnDemand to call the DLLs integrated into the client. Figure 3 on page 284 shows an example of a subkey with which a user can invoke the new model Monarch DLL from the client. Since the subkey doesn't specify the name of a model file to run, OnDemand simply starts Monarch with the current document.

Note: When a user invokes Monarch from the client and no model file is specified in the Registry key, OnDemand starts Monarch with the current document. The user can then use Monarch functions to analyze the data and optionally create and save a model. To subsequently run a model that was created and saved from the client, the model must be identified in Dlls and a subkey for the model must be added under ExternalDlls. The subkey

for the model must specify the values that are required to run the model from the client. For example, the Parameter value must contain the full path name of the model file and any Monarch parameters that are required to run the model.

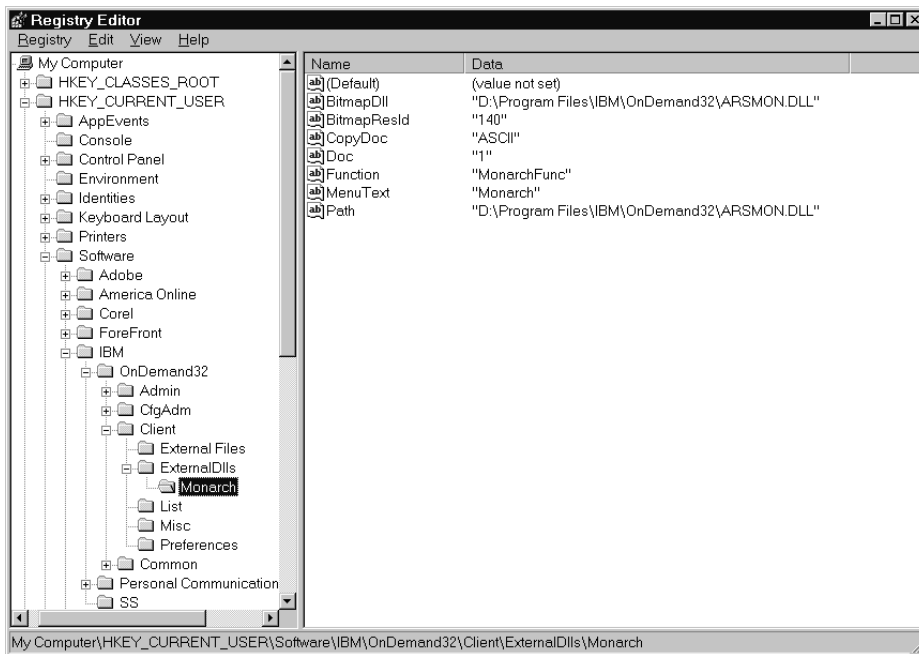


Figure 3. Subkey and values for the Monarch DLL

Table 2 provides information about the subkey values and value data shown in Figure 3.

Table 3 on page 286 provides information about other values (that are not depicted in Figure 3) that you may specify.

Table 2 (Page 1 of 2). Monarch DLL subkey values

Value and Data	Description
BitmapDll D:\Program Files\IBM\OnDemand32\ARSMON.DLL	The full path name of the DLL file that contains the bitmap resource for the toolbar button used to invoke Monarch from the OnDemand client. The example uses the default provided by IBM. However, this DLL file may be different from the DLL file specified for the Path or you can use a different BitmapDll DLL file. The bitmap should be 16 picture elements (pel) wide and 16 pels high. This value is optional. If not provided, a toolbar bitmap will not be created.
BitmapResId 140	The resource identifier of the bitmap within the DLL file specified for BitmapDll. This value is ignored if BitmapDll is not specified and optional if it is. If not provided, a value of 0 (zero) is assumed.

Table 2 (Page 2 of 2). Monarch DLL subkey values

Value and Data	Description
CopyDoc	Indicates that a copy of one or more documents is to be provided to Monarch and specifies the type of data to be provided. If the value is ASCII, the document data is converted to an ASCII file. If the value is ASIS, the document data is in its native format.
ASCII	<p>If the user chooses the menu item or toolbar button when a document is being viewed, the data provided is for the current document. If the document list is being displayed, the data is a concatenation of all documents selected in the document list.</p> <p>This value is optional. If not provided, no document is provided to Monarch.</p>
Doc	May be specified as one of the following values:
1	<p>0 (zero) indicates that enabling the associated menu item and corresponding toolbar button is limited only by the Folders and ExcludeFolders values. See for information about the Folders and ExcludeFolders values.</p> <p>1 (one) indicates that the menu item and toolbar button are enabled only when a document is being viewed.</p> <p>2 (two) indicates that the menu item and toolbar button are enabled only when the document list is being viewed and at least one document is selected.</p> <p>3 (three) indicates that the menu item and toolbar button are enabled only when a document is being viewed or a document list is being viewed and at least one document is selected.</p> <p>This value is optional. If not provided, a value of 3 (three) is assumed.</p>
Function	The name of the entry point into the Monarch DLL file. This value must be provided.
MonarchFunc	
MenuText	The text that appears on the menu item used to invoke Monarch from the OnDemand client. The menu item appears under the Window menu. The text also appears when the user passes the mouse pointer over the associated toolbar button. note. The text you specify can describe a specific application in Monarch, such as Loan Analysis.
Monarch	This value is optional. If not provided, the menu item is blank.
Path	The full path name of the Monarch DLL file on the user's PC. The example shows the default installation drive and directory. This value must be provided.
D:\Program Files\IBM\OnDemand32\ARSMON.DLL	

Table 3. Other DLL subkey values

Value	Description
ExcludeFolders	<p>Specified the names of one or more OnDemand folders. The syntax is the same as the Folders value.</p> <p>The menu item and toolbar button are enabled whenever:</p> <ul style="list-style-type: none"> • A document is being viewed and the folder associated with the document is not one of the specified folders, or • A document is not being viewed and the current folder is not one of the specified folders. <p>This value is optional. If not provided, enabling is controlled by the Folders value.</p>
Folders	<p>Specifies the names of one or more OnDemand folders. If you specify more than one name, you must separate the names with the backslash (\) character. An asterisk (*) character may be used as a wildcard character in the last position of the name. (This is equivalent to listing all of the folder names that begin with the characters preceding the asterisk.)</p> <p>The menu item and toolbar button are enabled whenever:</p> <ul style="list-style-type: none"> • A document is being viewed and the folder associated with the document is one of the specified folders, or • A document is not being viewed and the current folder is one of the specified folders. <p>This value is optional. If the ExcludeFolders value is provided, this value is ignored. If neither is provided, a folder name test is not performed before enabling the menu item and toolbar button.</p>
Parameter	<p>Specifies a maximum of 255 characters to be passed as parameter data to Monarch. This value is optional. If not provided, then no parameter data is passed to Monarch. See your Monarch information for a list of the parameters that you can specify.</p> <p>Note: To run a specific Monarch model, the Parameter value must contain the full path name of the model file and any Monarch parameters that are required to run the model. If you do not specify the Parameter value, then OnDemand simply starts Monarch with the current document. The user can then use Monarch functions to analyze the data and optionally create and save a model.</p>

To add the Registry key:

- The ExternalDlls subkey contains a list of the DLLs that are integrated with the OnDemand client. It also contains subkeys that define the properties of each DLL. Add the ExternalDlls subkey to the following subkey:

HKEY_CURRENT_USER\Software\IBM\OnDemand32\Client

- Add the value Dlls to the ExternalDlls subkey. Dlls must have a type of String (REG_SZ). The data value of Dlls is a comma-separated list of identifiers for the DLLs that are integrated with the OnDemand client. Each identifier in the list must have an associated subkey under ExternalDlls. In , the identifier is Monarch.
- Define each DLL listed in Dlls by adding a subkey under the ExternalDlls subkey. (Each subkey that you add must be listed in Dlls.) In , we added the Monarch subkey.

- Define the properties of a DLL by adding string values to the subkey that you added. In the example (see), we added seven string values. To run a specific Monarch model, the Parameter subkey value must contain the full path name of the model file and any Monarch parameters that are required to run the model.

Exporting the Registry key

After you have added the Registry key that defines the properties of the Monarch DLL to the OnDemand Windows 32-bit client (and tested that you can invoke Monarch from the client), you can export it to a file. Later, you will store the file in the CUSTOM subdirectory tree under the OnDemand Windows 32-bit client installation directory tree on a distribution server. Once you have placed the file there, users installing the OnDemand client will automatically have the Registry key imported into the Registry on their PC when they run Setup from the distribution server. If you have questions about the Registry key, see “Adding the Registry key” on page 283.

Important: If you have other applications integrated with OnDemand (that is, you have DLLs from applications other than Monarch defined under the ExternalDlls subkey), contact the IBM support center before you proceed.

To export the Registry key, do the following:

1. Start the REGEDIT program.
2. Move to the HKEY_CURRENT_USER\IBM\OnDemand32\Client\ExternalDlls key.
3. From the Registry menu, select Export Registry File.
4. Select a directory to hold the file.
5. In the File Name field, enter: ODMonarch.Reg
6. Under Export Range, select Selected Branch.
7. Click Save.

Using multiple Monarch model files

“Adding the Registry key” on page 283 describes how to configure the client on the reference workstation to run a single instance of Monarch. You can configure the client to run multiple instances of Monarch, with each instance using a different model file. To configure the client, add an identifier for each instance of Monarch to the Dlls string value and add a subkey under the ExternalDlls subkey for each identifier. Each subkey should represent a different model file that you want to let your users run from the client. This section provides information to help you configure the client with multiple model files.

For each model file that you want to run from the client:

- Add an identifier to the Dlls string value under the ExternalDlls subkey. Add a unique identifier for each model file. Figure 4 on page 288 shows an example of the ExternalDlls subkey with multiple identifiers. (We added the identifiers Model1, Model2, and Model3.)

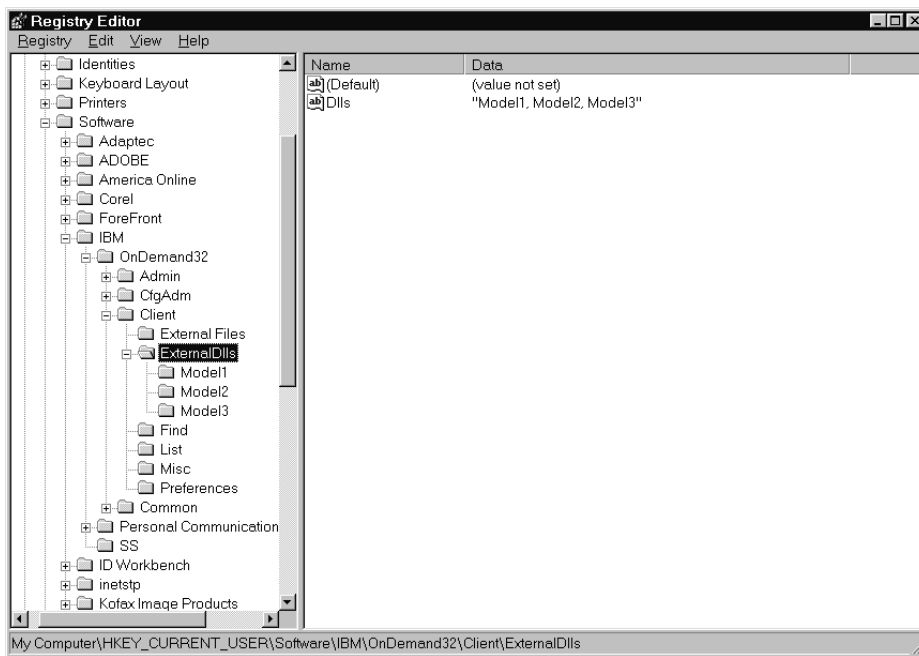


Figure 4. ExternalDlls subkey showing multiple values in Dlls

- Define the DLL to the client by adding a subkey under the ExternalDlls subkey. Each subkey that you add must be listed in Dlls. Define the properties of the DLL by adding string values to the subkey that you added:
 - Use the Parameter value to specify the full path name of the model file. To run a specific Monarch model, the Parameter value must contain the full path name of the model file and any Monarch parameters that are required to run the model. See your Monarch information for a list of the parameters that you can specify.
 - You can use the MenuText value to provide specific menu text and hover help for each model file
 - You can use the BitMapDll value to specify a different toolbar button bitmap for each model file

See Table 2 on page 284 and Table 3 on page 286 for help with the properties that you can specify.

When you have finished defining the model files to the client, export the Registry key. Use the process described in “Exporting the Registry key” on page 287.

Configuring Setup

This topic describes how to configure the client installation program to copy your Monarch model files to the user's PC and import the Registry key into the Registry on the user's PC. To begin, extract the latest version of the OnDemand Windows 32-bit client software from the ZIP file to a directory on the distribution server. Then add the custom directories to the OnDemand Windows 32-bit client installation directory tree on the distribution server. Next, copy the Monarch files to the custom directories. Then share the installation directory tree to make it available to your users. The following topics provide more information:

- Copying client software to the distribution server
- Adding custom directories
- Copying the Monarch files to the distribution server
- Sharing the installation directory

Copying client software

Note: See “Before you begin” on page 282 for information about obtaining the latest OnDemand Windows client software.

To copy the client software to the distribution server:

1. Log on to the server with a userid that has administrator permissions.
2. Extract the contents of the OnDemand Windows 32-bit client ZIP file (odwin32.zip) to a directory on the server. For example, extract the files to the \OD2217 directory.

Important: Use a file extraction method that preserves the ZIP contents directory and file structure on the server.

When complete, the target directory (\OD2217 in the example) should contain the Setup program and files and the ARS directory tree.

Adding subdirectories

You must store the Monarch files in the CUSTOM subdirectory tree under the OnDemand Windows client directory tree on the distribution server. By default, the directory tree is ARS32. (If you followed the example in “Copying client software,” the client directory tree on the distribution server will be \OD2217\ARS32.) To add subdirectories to hold the Monarch files:

1. Create a CUSTOM directory under the Windows client directory. For example:

```
mkdir \od2217\ars32\custom
```

2. Add the \od2217\ars32\custom\registry directory.

This directory will hold the Registry file (ODMonarch.Reg) that you created in “Exporting the Registry key” on page 287. When the user runs the Setup program from the distribution server, it imports this file into the Registry on the PC.

3. Add the \od2217\ars32\custom\monarch directory.

This directory will hold the Monarch model files that you need to distribute to your users. The Setup program copies the Monarch model files to the Monarch\Models directory on the user's PC.

Copying Monarch files

After copying the OnDemand client software to the distribution server and creating the CUSTOM directories (see “Adding subdirectories” on page 289), copy the Monarch files to the subdirectories. For example:

- Copy the ODMonarch.Reg file that you created in “Exporting the Registry key” on page 287 to the \OD2217\ARS32\CUSTOM\REGISTRY directory
- Copy your Monarch model files to the \OD2217\ARS32\CUSTOM\MONARCH directory

Sharing the installation directory

After you copy the OnDemand client software to the distribution server, create the CUSTOM directories, and copy the Monarch files to the server, you must make the installation directories on the server available to your users. The procedure differs for each network and operating system. However, you generally need to provide users with read-only access to the directories. If applicable for your network, share the folders by giving the folder location on the distribution server a network name (share name). For example, to permit users to run the Setup program from the \OD2217 directory on the server and access all of the files in the \OD2217\ARS32 directory tree, you could assign the share name ODMONAR to the \OD2217 directory.

Running Setup

Note: Before your users run the Setup program from the distribution server to install the OnDemand client and integrate Monarch with the client, they must have a copy of Monarch Version 5 installed on their PCs.

After you have configured the Setup program on the distribution server with the Registry key and your Monarch model files, you can test the installation by having a user run the Setup program from the distribution server.

Note: It is easy to accidentally destroy important data and render a system completely unusable by importing Registry files. Plan to backup the Registry on the user's PC before they run the Setup program.

When the user selects the Typical or Compact option during install, the Setup program automatically copies the Monarch files to the PC.

When the user selects the Custom option during install, the Setup program copies the Monarch files to the PC if the user selects to install one of the clients. If the user does not select one of the clients, then the Setup program does not copy the Monarch files to the PC. For example, if the user selects the Custom option and chooses to install only the Sonoran Fonts or the administrative client, the Setup program does not copy the Monarch files to the PC; if the user selects the Custom option and chooses to install the US English client, then the Setup program copies the Monarch files to the PC.

Running Monarch from OnDemand

After your users install the client from the distribution server, they should test that everything works correctly. Have them start the OnDemand client, open a document, and then start Monarch using the associated menu command or toolbar button. Monarch should start and load the document into a separate window.

Upgrading your OnDemand client

If your organization has integrated Monarch with OnDemand Version 2.2.1.2 or earlier, follow these important instructions before your users upgrade the client to OnDemand Version 2.2.1.3 or later:

- Copy your old Registry file (for example, ODMonarch.Reg) to the CUSTOM\REGISTRY directory on the distribution server.
If the old Registry file does not exist, follow the steps in to create one. Then copy the Registry file to the CUSTOM\REGISTRY directory on the distribution server.
- Copy your Monarch model files to the CUSTOM\MONARCH directory on the distribution server.

Installing client software on a network

This section provides information about installing the client software to be shared by multiple users over a network. This section is of primary interest to administrators responsible for installing and configuring software products.

Sharing OnDemand clients among multiple users

There are several ways that you can install OnDemand client software:

Standard or Custom Install

Use to install the OnDemand client to a hard disk on the user's PC. These users do not have to rely on a network file server to run the software. To run a standard install, start the Setup program and select the Typical option. A custom install is required if you need to install the administrative client on a PC, if you need to install the OnDemand client in a language other than the default language set on the PC, or if you need to install the Sonoran Fonts on a PC. To run a custom install, start the Setup program and select the Custom option. Then select the components to install. The *User's Guide* describes the standard and custom installation procedures.

Distribution install

Use to copy the OnDemand client software to a hard disk on a network file server. Users can then use the copy on the server to perform subsequent standard, custom, distribution, multiple user, and node installs. A distribution install is also useful if you need to add user-defined files to the client installation. "Distribution install" on page 294 describes the distribution installation procedure.

Multiple User Install

Use to install a copy of the OnDemand client to a hard disk on a network file server. Users can then run a Node Install on their PC to configure the PC to run the client from the server. "Multiple user install" on page 296 describes the multiple user installation procedure.

Node Install

Use to configure a user's PC to run a copy of the OnDemand client from a shared location on a network file server. To run a node install, select the Compact option from the Setup program. A node install copies OnDemand client control files to the user's PC; no program files are copied to the user's PC. The user runs a copy of the OnDemand client program from the network server. After completing the installation, when the user starts OnDemand, the operating system loads the OnDemand programs from the server into memory on the PC. OnDemand allocates temporary work space on the user's PC for data, resources, and printing. When the user views documents, OnDemand saves any client options the user

changes on the user's PC. A node install requires approximately 2.5 MB of disk space on the user's PC. [href refid=node.](#) describes the node installation procedure.

Installation directories

The OnDemand client CD-ROM is organized into directories for each of the clients. The Windows client comes with one program for the standard, custom, multiple user, and node installs:

`\client\win32\setup.exe`

When you install a client, the installation program copies the client software to the directories listed in Table 4.

Table 4. Windows client installation directories

Directory	Purpose
\Program Files\IBM\OnDemand32	Program directory. For standard, custom, and multiple user installs, identifies a hard disk on the PC. For node installs, identifies a network drive.
\Program Files\IBM\OnDemand32	Local directory. Identifies a hard disk on the PC.
\Program Files\IBM\OnDemand32\DATA	Temporary data directory. Identifies a hard disk on the PC.
\Program Files\IBM\OnDemand32\FONTS	AFP font file directory. For standard, custom, multiple user installs, identifies a hard disk on the PC. For node installs, identifies a network drive.
\Program Files\IBM\OnDemand32\PRINT	Temporary print directory. Identifies a hard disk on the PC.
\Program Files\IBM\OnDemand32\RES	Temporary resource directory. Identifies a hard disk on the PC.
\PSFONTS ³	Outline font directory. For standard, custom, multiple user installs, identifies a hard disk on the PC. For node installs, identifies a network drive.

Distribution install

³ Represents the directory in which the outline fonts are stored. If ATM is already installed on the PC, then the Setup program stores the outline fonts in the directory set by ATM. Otherwise, the Setup program stores the fonts in \IBM\OnDemand32\PSFONTS.

Overview

A distribution install copies the contents of the OnDemand Windows client directory tree from the OnDemand client CD-ROM to a shared location on a network file server. You can then use the copy on the network file server to perform subsequent standard, custom, distribution, multiple user, and node installs. The distribution install is also useful if you need to add user-defined files to the client installation.

Note: You can also distribute OnDemand client software by sharing the CD-ROM from the CD-ROM drive on a network file server.

Distributing Adobe software: If you need to distribute ATM or Adobe Acrobat viewing software to your OnDemand users, you can copy the Adobe software to a shared location on a network file server. Other users can then use the copy to install the Adobe software on their PCs. See the *User's Guide* for information about obtaining and installing Adobe software.

Copying OnDemand software to the server

To copy the OnDemand client software to a network file server, follow these steps:

1. Log on to the server with a userid that has administrator permissions.
2. Insert the OnDemand client CD-ROM in the drive.
3. Copy the entire contents of the \CLIENT\WIN32 client directory tree to a drive on the network file server.

Note: Use a copy method that preserves the directory and file structure of the CD-ROM on the network file server.

After you copy the OnDemand client software to the network file server, make the server directories (or folders) available to network users. The procedure differs for each network and operating system. However, you generally need to give users read-only access to directories. If applicable for your network, share the folders by giving the folder location on the network file server a network name (share name).

After you copy the software to the server and share the directories, users can run a standard or custom install from the server to install a client on their PC. An administrator can also run a multiple user install from the server to install a client to a different network location. Users can then run a node install to configure their PC to run the client from that location.

Distributing user-defined files

An administrator can store user-defined files in the OnDemand client installation directory tree on a distribution server. Any user-defined files stored there get copied to the PC when a user runs the Setup program from the server. User-defined files can be distributed to Windows

clients. To read more about support for user-defined files, see “Distributing user-defined files” on page 299.

Multiple user install

Overview

The first part of a multiple user install is to run a standard install of the OnDemand client. Use the standard install to install the OnDemand client to a shared location on a network file server. After installing the client on the server, other users on the network can run a node install to configure their PC to run the client from the network server. “Node install” on page 297 describes how to run a node install on a network user's PC.

Installing Adobe software

If your OnDemand users view documents that require Adobe Type 1 Fonts, ATM must be installed on each user's PC. OnDemand provides ATM software on the client CD-ROM. We recommend that you install ATM before you install the client software on a PC. The *User's Guide* describes how to install ATM.

To view PDF documents stored in OnDemand, users need Adobe Acrobat viewing software. You may want to install the Acrobat viewing software to a shared location on a network file server. That way, network users can run the Acrobat software from the server. See the *User's Guide* for information about obtaining and installing Adobe Acrobat software.

Installing the OnDemand client on the server

To install the OnDemand client on a network file server, follow these steps:

1. Log on to the server with a userid that has administrator permissions.
2. Insert the OnDemand client CD-ROM in the drive.
3. Start the OnDemand client installation program:

`\client\win32\setup.exe`

4. Follow the installation instructions on the screen.

Note: If your users need to run the administrative client, run the client in a language other than the default language set on the server, or use the Sonoran Fonts, then select the Custom option instead of the Typical option. The Custom option will allow you to select the additional components that you need to install on the server.

5. Verify the drives and directories. When you install the client on a network file server, all of the directories typically reside on a disk on the server. Table 5 on page 297 shows an example.

Table 5. Drives and directories on the network file server

Drive and Directory	Purpose
C:\APPS\ARS32	Destination Folder
C:\APPS\ARS32\PSFONTS	Outline Fonts Folder

After you install the OnDemand client on the network file server, make the server directories (or folders) available to network users. The procedure differs for each network and operating system. However, you generally need to give users read-only access to directories. If applicable for your network, share the folders by giving the folder location on the network file server a network name (share name). After you install the client on the server and share the directories, users can run a node install to configure their PC to run the client from the server. “Node install” describes how to run a node install.

Sharing user-defined files

An administrator can store user-defined files in the OnDemand client directory tree on a network file server. Any user-defined files stored there can be shared by the users that run the client from that location. User-defined files can be shared by Windows clients. To read more about support for user-defined files, see “Distributing user-defined files” on page 299.

Node install

Overview

A node install is used to configure a user's PC to run the OnDemand client from a shared location on a network file server. The following procedure shows how to run a node install from the OnDemand client CD-ROM. The procedure assumes that you have already installed the OnDemand client to a shared location on a network file server (see “Distribution install” on page 294 for details). You can also run a node install from a distribution server. See “Distribution install” on page 294 for information about copying OnDemand software to a distribution server.

Installing Adobe software

If your OnDemand users view documents that require Adobe Type 1 Fonts, ATM must be installed on each user's PC. OnDemand provides ATM software on the client CD-ROM. We recommend that you install ATM before you install the client software on a PC. The *User's Guide* describes how to install ATM. To view PDF documents stored in OnDemand, users need Adobe Acrobat viewing software. If you installed the Acrobat viewing software to a shared location on a network file server, users can run the software from that location. Otherwise, see the *User's Guide* for information about obtaining and installing Adobe Acrobat software on a PC.

Installing the client

To configure a user's PC to run the OnDemand client from a shared location on a network file server, follow these steps:

1. Insert the OnDemand client CD-ROM in the drive.
2. Run the OnDemand client installation program:

\client\win32\setup.exe

3. Follow the instructions on the screen.

Note: Most users should select the Compact option.

4. Verify the drives and directories. Make sure that the Network Program Folder and the Network Font Folder locations identify drives and directories on the network file server on which you installed the OnDemand client software (see “Multiple user install” on page 296). The Destination Folder must identify a drive and directory on the user's PC. Table 6 shows an example.

Table 6. Drives and directories on the network user's PC

Drive and Directory	Purpose
C:\Program Files\IBM\OnDemand32	Destination Folder; typically identifies a hard disk on the user's PC
N:\APPS\ARS32	Network Program Folder; identifies the location of the OnDemand Windows client program files on the network file server
N:\APPS\ARS32\PSFONTS	Network Font Folder; identifies the location of the OnDemand outline fonts on the network file server

Distributing user-defined files

This section provides information about how to configure the OnDemand Windows client installation program to distribute user-defined files to your users. This section is of primary interest to administrators responsible for installing and configuring software products.

Overview

User-defined files can be stored in the OnDemand Windows client directory tree on a distribution server.⁴ Any files (and subdirectories of files) that you store there are copied along with the standard OnDemand client files when a user installs the client from the server to a PC. You can distribute the following types of user-defined files:

- Adobe Type 1 Font files. These files are copied to the \PSFONTS⁵ directory tree on the user's PC.

Note: The installation program copies Type 1 Font files to the user's PC. The user must install the fonts with the ATM applet in Control Panel.

- Registry files. These files are imported into the Registry on the user's PC.

Note: It is easy to accidentally destroy important data and render a system completely unusable by importing Registry files. Be careful when using this function.

- Windows files. These files are copied to the base Windows directory tree on the user's PC.
- Miscellaneous client files. These files are copied to the \Program Files\IBM\OnDemand32 directory tree on the user's PC. You can use this feature to replace (overwrite) files supplied by IBM. When a user runs the Setup program from the server, it first copies the files supplied by IBM to the PC. The Setup program then copies any user-defined files to the PC. If a user-defined file has the same name as a file supplied by IBM, the IBM-supplied file will be overwritten by the user-defined file. For example, suppose you need to modify the AFP character set definition file (CSDEF.FNT) to map fonts your documents were created with to fonts that can be displayed on the PC. You can automatically distribute an updated version of the file to your users by storing the modified CSDEF.FNT file in the CUSTOMFILES\FONT directory on the distribution server. When a user installs the client from the server, the Setup program first copies the

⁴ A distribution server is a network file server that contains a copy of the OnDemand client software in a shared location. Other users on the network use the copy to run standard, custom, distribution, multiple user, and node installs. See "Installing client software on a network" on page 293 for more information about setting up a distribution server to share OnDemand software on a network.

⁵ Represents the directory where the font files are stored. If ATM is already installed on the PC, the Setup program stores the font files in the directory set by ATM. Otherwise, the Setup program stores the font files in the \Program Files\IBM\OnDemand32\PSFONTS directory.

CSDEF.FNT file supplied by IBM to the PC. The Setup program then copies the CSDEF.FNT file you modified to the PC.

The following topics contain more information about distributing user-defined files:

- Copy OnDemand client software to a distribution server
- Add subdirectories to hold user-defined files
- Store user-defined files in subdirectories
- Install the OnDemand client on a user's PC

Copying OnDemand client software to the server

See “Installing client software on a network” on page 293 for instructions about copying the OnDemand client software to a distribution server.

Adding subdirectories

All user-defined files must be stored in the CUSTOM subdirectory tree under the OnDemand Windows client directory tree on the distribution server. By default, the Windows client directory tree is ARS32. (If you followed the instructions in “Installing client software on a network” on page 293, then the OnDemand Windows client directory tree on the distribution server will be \CLIENT\WIN32\ARS32.)

To configure the server to install user-defined files:

1. Create a CUSTOM directory under the Windows 32-bit client directory. For example:

```
mkdir \client\win32\ars32\custom
```

2. Add one or more of the following subdirectories to the CUSTOM directory. The subdirectories that you add will depend on the type of user-defined files you want to distribute to your users. For example:

\client\win32\ars32\custom\psfonts

To hold Adobe Type 1 Font files (file type PFB). If necessary, add a PFM subdirectory (\CLIENT\WIN32\ARS32\CUSTOM\PSFONTS\PFM) to hold the PFM files. The Setup program copies these files and subdirectories to the \PSFONTS directory tree on the user's PC.

Note: The installation program copies Type 1 Font files to the user's PC. The user must install the fonts with the ATM applet in Control Panel.

`\client\win32\ars32\custom\registry`

To hold Registry files (file type REG). These files will be imported into Registry on the user's PC. Registry files typically comprise a selected branch of the Registry exported using a Registry editor.

Note: It is easy to accidentally destroy important data and render a system completely unusable by importing Registry files. Be careful when using this function.

`\client\win32\ars32\custom\windows`

To hold Windows files. The Setup program copies these files and subdirectories to the base Windows directory on the user's PC. (The Setup program automatically determines the name of the base Windows directory on a user's PC.) If necessary, add subdirectories. For example, suppose you plan to distribute Window 32-bit system files. You would add a SYSTEM32 subdirectory (`\CLIENT\WIN32\ARS32\CUSTOM\WINDOWS\SYSTEM32`). The Setup program copies files in this directory to the `\WINDOWS\SYSTEM32` directory on the user's PC.

`\client\win32\ars32\custom\files`

To hold miscellaneous OnDemand client files. The Setup program copies these files and subdirectories to the `\Program Files\IBM\OnDemand32` directory tree on the user's PC. If necessary, add subdirectories. For example, you plan to distribute AFP font files. Add a FONT subdirectory (`\CLIENT\WIN32\ARS32\CUSTOM\FILES\FONT`). The Setup program copies files in this directory to the `\Program Files\IBM\OnDemand32\FONT` directory on the user's PC.

Storing user-defined files on the server

After copying the OnDemand client software to the distribution server and creating the CUSTOM directories, store files in the individual subdirectories. For example, store Adobe Type 1 Font files (PFB) that you want to be copied to a user's PC into the `\CLIENT\WIN32\ARS32\CUSTOM\PSFONTS` directory. Store the PFM files in the `\CLIENT\WIN32\ARS32\CUSTOM\PSFONTS\PFM` directory.

Installing the OnDemand client

After you set up the CUSTOM directory tree on the distribution server, users can begin installing the client and the user-defined files. The next time that a user runs the Setup program from the server, the Setup program installs the OnDemand client on the PC and copies all of the user-defined files that you stored on the server to the user's PC.

When a user selects either the Typical or Compact option during install, the Setup program automatically copies the user-defined files to the PC.

When a user selects the Custom option during install, the Setup program may or may not copy the user-defined files to the PC. If the user chooses to install one of the clients, then the Setup program copies the user-defined files to the PC. Otherwise, the Setup program does not copy the user-defined files to the PC. For example, if the user selects the Custom option and chooses to install only the Sonoran Fonts, then the Setup program does not copy the user-defined files to the PC. However, if the user selects the Custom option and chooses to install the administrative client, then the Setup program copies the user-defined files to the PC.

Note: If you distributed Adobe Type 1 Font files to your users, they must install the fonts with the ATM applet in Control Panel after the OnDemand client installation program completes.

Using response files

This section describes how to use response files to automate the installation of the client software. This section is of primary interest to administrators responsible for installing and configuring software products.

Introduction

This section provides information that can be used to create and use response files to install the Windows client software on PCs connected to the network. You typically use the Setup program to create a response file. You then install the software on other PCs by running the Setup program and specifying the name of the response file.

A response file is an ASCII file that supplies the client-specific configuration information required during redirected installation of a product on a PC. The response file contains predefined answers to the configuration questions that users are normally asked during a product installation, such as the installation drive and directory and the components to install. A system administrator can use a response file to automate the installation and configuration of the Windows 32-bit client software over a network of PCs. The response file makes it unnecessary for the system administrator (or other user) to sit at each PC and manually enter an answer to each question that is displayed during installation.

Format of a response file

The format of a response file is similar to that of an .INI file. A response file contains pairs of keywords and values organized into sections. The keywords and values are interpreted during software installation.

Creating a response file

Response files commonly have an extension of .ISS and are found in the Windows directory.

You can create a response file by running the Setup program with the -r command line option. For example, the command:

```
\client\win32\setup -r
```

Causes the Setup program to record all of your answers to the product installation questions in the SETUP.ISS response file. You can direct the Setup program to place the response file in a different directory and name a response file by specifying the -f1 command line option. For example, the command:

```
\client\win32\setup -r -f1n:\client\win32\ars32in.iss
```

Causes the Setup program to create the ARS32IN.ISS file in the \CLIENT\WIN32 directory on the N drive.

Installing software using a response file

A response file is not invoked directly. Instead, a response file is specified as a parameter value for the installation program. You can run the Setup program and specify a response file with the -s command line option. For example, the command:

```
\client\win32\setup -s
```

Causes the Setup program to install the software using the instructions found in the SETUP.ISS response file in the \CLIENT\WIN32 directory on the current drive. By default, the response file must be located in the directory from which you run the Setup program. Use the -f1 option to identify the location and name of the response file. For example, the command:

```
\client\win32\setup -s -f1n:\client\win32\ars32in.iss
```

Causes the Setup program to install the software using the ARS32IN.ISS response file located in the \CLIENT\WIN32 directory on the N drive.

The response file directs the processing of the installation for the Windows 32-bit client software. When you run the Setup program with the -s option, no messages or dialog boxes are displayed. Instead, messages are written to a log file. By default, the log file (SETUP.LOG) is written to the directory from that contains the Setup program. You can direct the Setup program to place the log file in a different directory and name the log file by specifying the -f2 command line option. For example, the command:

```
\client\win32\setup -s -f1n:\client\win32\ars32in.iss -f2c:\temp\ars32in.log
```

Causes the Setup program to write the log file ARS32IN.LOG in the TEMP directory on the C drive.

Verifying software installation

To verify the installation of a product that you installed using a response file, open the log file and locate the ResponseResult section. Examine the value of the ResultCode parameter. The return code should be zero (0).

Using a response file to install OnDemand software

In general, you would complete the following steps to prepare the OnDemand Windows client software for installation using a response file and then install the software on other PCs connected to the network.

1. Install the software on a PC. Run the Setup program with the -r option to create the response file and the -f1 option to identify the location and name of the response file. When prompted by the Setup program, select the Typical option.
2. Test the installation process and the response file by installing the software on a user's PC. Specify the name of the response file you created in step .
3. After testing and validating the response file, install the software on other PCs. Run the Setup program with the -s option to read the response file you created in step , the -f1 option to identify the response file, and the -f2 option to identify the directory where the Setup program writes the log file.
4. Examine the log files to verify the installation of the software.

Mapping AFP fonts

The OnDemand client needs to map the AFP fonts your document was created with to fonts that can be displayed on your workstation. For the client to map the best matching outline fonts to display AFP documents, it needs to know certain characteristics about the fonts that were used to create the documents. Mapping AFP fonts to outline fonts is done with the IBM-supplied font definition files that are installed as part of the client. These files are installed into the FONT directory under the directory in which you installed the client. You may edit them using any workstation editor. The installed version of the font definition files maps the IBM Core Interchange (Latin only), compatibility, coordinated, Sonoran, and Data1 fonts for you.

If your document uses an AFP font whose family (familyname) is not installed on your workstation, then you can use the ALIAS.FNT file (one of the font definition files installed with the client) to substitute that font familyname with a different one. The ALIAS.FNT file remaps several of the AFP fonts to IBM Core Interchange fonts. If you have any outline fonts installed on your workstation, you may want to remove or comment out the font familyname substitutions in the ALIAS.FNT file. See “Alias file” on page 318 for more information about using the ALIAS.FNT file.

The IBM Core Interchange fonts (provided with the OnDemand client) are in Type 1 outline format. These fonts are delivered in three type families: Times New Roman, Helvetica, and Courier. Each type family is provided in these character set groups:

Latin The Latin group is available in 4 typefaces: roman medium, roman bold, italic medium, and italic bold.

Symbols The Symbols group is available in 2 typefaces: roman medium and roman bold.

Because the IBM Core Interchange fonts are also available for printing with Infoprint, they help standardize fonts across applications and installations.

If you created your documents with only the unmodified IBM fonts, then you do not need to remap fonts to view them correctly.

When you need to map fonts

If you are using fonts that are not defined to OnDemand, if you have modified the IBM AFP fonts, or if you have created your own AFP fonts (for example, with PSF/2 Type Transformer), then you need to define those fonts in the font definition files in order for documents using those fonts to display correctly with the client.

- If you created a new coded font (or renamed one), you will need to define the coded font in the Coded Font file (ICODED.FNT or CODED.FNT).

- If you created a new character set, you have to define it in the Character Set Definition file (CSDEF.FNT).
- If you created a new code page, you have to define it in the Code Page Definition file (CPDEF.FNT).
- If you have created a new code page or modified a code page by moving characters, you have to create a new Code Page Map file (cp_id.CP).

If you only have modified an existing IBM font component, you may not need to perform any of the above steps. For example, if you have only deleted code points in the IBM code page, then the font files supplied with the client can be used.

Files supplied for mapping fonts

The following types of files for font support are installed by default in the following subdirectories under the directory in which the client was installed:

File	File Name	Subdirectory	Description
Coded Font files	CODED.FNT ¹ ICODED.FNT ICODED.CHS ² ICODED.CHT ³ ICODED.JPN ⁴ ICODED.KOR ⁵	..\FONT	Specifies which AFP code page and AFP font character set make up the coded font.
Character Set definition file	CSDEF.FNT CSDEF.CHS ² CSDEF.CHT ³ CSDEF.JPN ⁴ CSDEF.KOR ⁵	..\FONT	Defines AFP character set attributes, such as point size. It also maps the font character set to its font global identifier.
Code Page definition file	CPDEF.FNT CPDEF.CHS ² CPDEF.CHT ³ CPDEF.JPN ⁴ CPDEF.KOR ⁵	..\FONT	Maps each AFP code page to a Windows character set ⁶ and indicates which Code Page Map file for the client to use.
Code Page Map file	<i>cpgid.CP</i>	..\FONTMAPS	Defines character identifier mappings. It matches the IBM code page character identifiers and their hexadecimal code points with a corresponding character identifier and ASCII code point representing a Windows ANSI or SYMBOL character set. ⁶

File	File Name	Subdirectory	Description
Alias File	ALIAS.FNT	..\FONT	Maps AFP font type families to Type 1 or TrueType outline font family names.
Notes: <ol style="list-style-type: none"> 1. CODED.FNT is an optional file. A sample can be found in the SAMPLES subdirectory of the FONT subdirectory. The CODED.FNT file is meant to contain coded fonts that you created. 2. Code page and character set files for the Simplified Chinese client. 3. Code page and character set files for the Traditional Chinese client. 4. Code page and character set files for the Japanese client. 5. Code page and character set files for the Korean client. 6. The Windows term “character set” is roughly equivalent to the AFP term “code page.” 			

Table 7. Font files and directories

Steps for mapping fonts

After reading the rest of this chapter to determine which font files you need to modify, follow these steps:

1. Gather the information needed to define the fonts in the font definition files. This information is described in the following sections of this appendix.
2. Make backup copies of any of the following font definition files that you plan to modify:
 - CSDEF.FNT
 - CPDEF.FNT
 - ICODED.FNT
 - ALIAS.FNT

Note: Backup copies of these files should be made so that you have an unmodified copy in the event something happens to your modified copy that makes it inoperable.
3. Install any other outline fonts you are planning to use with the client. (See *Adobe Type Manager User Guide* for information on installing fonts with ATM.)
4. If you have created or modified a code page, use the BLDCPMAP REXX program to build the code page map file:
 - a. Determine which Windows character set (ANSI or SYMBOL) is a suitable match for the AFP code page.
 - b. Substitute any non-matching characters in the code page map file or ALIAS.FNT file if necessary. (See “Code Page Map files” on page 315 and “Code Page Map file REXX program for building a Code Page Map file” on page 317 for information about code page map files and the code page map file program respectively.)
 - c. Edit the CPDEF.FNT file and add your code page name, code page identifier, and the best matching Windows character set name for the fonts you are using.

Note: If you are specifying the SYMBOL Windows character set, the font familyname used with the code page must be a symbol font.

5. If you have created a new character set, edit the CSDEF.FNT file and add your character set name in the [CHARSET] section. Specify the correct attributes for your font in the CSDEF.FNT. Add the appropriate information in the [FGID] section of the file if you are naming a new font global identifier.
6. If you have created a coded font, create or edit the CODED.FNT file and add your coded font.

Syntax rules for font definition files

Syntax rules for the font definition files are as follows:

- A semicolon (;) in the first column of any of these files will cause the line to be treated as a comment statement and ignored.
- Section headers within files are enclosed in brackets [] and must *not* be removed or changed.
- All values are case insensitive.
- If a parameter value is invalid and a default value exists, it will be substituted.
- All parameters are positional.
- Blanks are allowed between parameter values.

Coded Font file

The IBM Coded Font file (ICODED.FNT) maps AFP coded fonts to their AFP character sets and AFP code pages. Two Coded Font files can be used with the client:

ICODED.FNT This file contains definitions for approximately 2500 IBM-supplied coded fonts.

CODED.FNT You can create this optional file to define a list of any coded fonts you have created. If you create a CODED.FNT file, you must place it in the FONT subdirectory. A sample of this file can be found in the SAMPLES subdirectory of the FONT directory.

If a CODED.FNT file exists in the FONT subdirectory, it is searched first for the coded fonts used in an AFP file. If the coded font name is not found in CODED.FNT or if CODED.FNT does not exist, only the ICODED.FNT file supplied with the client will be searched.

```

X?A155N2 = C?A155N1, T1DCDCFS
X?AE10   = C?S0AE10, T1S0AE10
X?GT10   = C?D0GT10, T1D0BASE
X?ST15   = C?D0ST15, T1D0BASE
X?A0770C = C?A07700, T1DCDCFS
X?A0770I = C?A07700, T1GI0361
X0T0550C = C0T05500, T1DCDCFS

```

Figure 5. Example of the partial contents of a CODED.FNT file

Coded Font file rules

- A question mark (?) can be used as the wild-card character only for the second character in the coded font name and the character set name. This allows all the character rotations of the coded fonts to be handled with one entry for searching.

Note: A sequential search is performed for the coded font, and the first match is used (including the wild-card character).

- After the coded font name, the character set name must be listed first, followed by the code page name.
- The character set and code page *must* be separated by a comma.

Character Set Definition file

The Character Set Definition file specifies the character set attributes and font global identifier of the font. It is split into 2 sections, one for character sets [CHARSET] and one for font global identifiers [FGID].

```

[CHARSET]
;charset = fgid, height, width, strikeover, underline
C?H200A0=2304,110,73,0,0
C?H200D0=2304,140,93,0,0
C?N200B0=2308,120,80,0,0
C?4200B0=416,120,144,0,0
C?D0GT15=230,80,96,0,0
C?A155A0=33207,110,73,0,0
C?A175A0=33227,110,73,0,0
C?T055D0=4407,140,93,0,0
C?T17500=4555,100,67,0,0
C?T17560=4555,60,40,0,0
DEFAULT =2308,80,0

```

Figure 6. The [CHARSET] section. Example of the character set [CHARSET] section in the Character Set Definition file (CSDEF.FNT).

The first section identified by the section header [CHARSET] lists each AFP font character set and its corresponding attributes:

- Font global identifier (fgid)
- Font height
- Font width
- Strikeover
- Underline

Attribute	Possible Values	Shipped Default	Description
Fgid	IBM-defined FGID or your own defined FGID within this range: 3840 to 4095 or 65260 to 65534	2308	A unique value that identifies the type family, typeface, and sometimes the point size of the character set.
Height	1 to 990	80	The vertical size of the character set (minimal baseline-to-baseline value) expressed in tenths of a point. For example, a 9-point font would have a height of 90.
Width	0 to 99 (currently ignored)	0	The average horizontal size of the characters in 1440th of an inch. Currently, 0 is always used because Windows determines an appropriate font width based on the font height.
Strikeover	1 (means yes), 0 (means no)	0	A font whose characters all have a line, parallel to the character baseline, placed over the middle of the character.
Underline	1 (means yes), 0 (means no)	0	A font whose characters all have a line, parallel to the character baseline, placed under the character.

Table 8. Character Set Definition file attribute values for [CHARSET]

The second section, identified by the section header [FGID], lists each font global identifier and its corresponding attributes:

- Font type families
- Style
- Weight
- Italic

```

[FGID]
;fgid = familyname, style, weight, italic
230=Gothic,MODERN,MED,0
416=Courier,MODERN,MED,0
2304=Helvetica,SWISS,MED,0
2308=TimesNewRoman,ROMAN,MED,0
4407=SonoranSerif,ROMAN,MED,0
4555=SonoranSerif,ROMAN,BOLD,1
33207=SonoranSansSerif,SWISS,MED,1
33227=SonoranSansSerif,SWISS,BOLD,1

```

Figure 7. The [FGID] section. Example of the font global identifier [FGID] section in the Character Set Definition file (CSDEF.FNT).

Attribute	Description	Possible Values	Shipped Default
Familyname ¹	An outline font name or an AFP type family name. This name appears on the ATM Control Panel if you have the font installed on your workstation.	Any Adobe Type 1 font name or AFP type family name	TimesNewRoman
Style ²	The same as a Windows “family.” It is approximately equivalent to type family plus typeface style in AFP fonts.	SWISS, ³ ROMAN, ⁴ SCRIPT, ⁵ MODERN, ⁶ DISPLAY ⁷	ROMAN
Weight	The degree of boldness of a typeface caused by different thickness of the strokes that form a graphic character.	LIGHT, MED, BOLD	MED
Italic	A font whose characters slant to the right.	1 (means yes), 0 (means no)	0
Note: <ol style="list-style-type: none"> 1. “Familyname” is the same as “type family” in AFP fonts and “typeface name” in Windows. 2. “Style” is the same as Windows “family” and is roughly equivalent to “typeface style” and “type family” in AFP fonts. 3. SWISS is a proportionally spaced font, without serifs. 4. ROMAN is a proportionally spaced font, with serifs. 5. SCRIPT is a fixed-pitch font designed to look like handwriting. 6. MODERN is a fixed-pitch font, with or without serifs. 7. DISPLAY is a decorative font. 			

Table 9. Character Set Definition file attribute values for [FGID]

Character Set Definition file rules

- Parameters must be separated by a comma. Table 8 on page 312 and Table 9 list the possible values, and shipped default values for each parameter.
- In the [CHARSET] section of the file, only `fgid` and `height` (point size) are required.
- In the [FGID] section of the file, only the type `familyname` and `style` are required.
- A question mark (?) can be used as the wild-card character only for the second character in the character set name. This allows all the character rotations of the coded fonts to be handled with one entry while searching.

Note: A sequential search is performed for the character set, and the first match is used (including the wild-card character).

- The [CHARSET] section must come before the [FGID] section.
- You can set a default character set. The default character set that is defined in the file must be the last entry in the [CHARSET] section.
- If you add your own AFP font character set to the [CHARSET] section, you must assign it a font global identifier. Font global identifiers you create must be in the ranges of 3840 to 4095 or 65260 to 65534. If the new character set has the same `familyname`, `style`, `weight`, and `italic` attributes as an existing character set, you may use the same font global identifier; otherwise, you must add a unique font global identifier to the [FGID] section.

Code Page Definition file

The Code Page Definition file maps the IBM AFP code page name to its code page global identifier (CPGID) and to a Windows character set. The section header [CODEPG] is followed by a list of AFP code pages and their parameters. The first parameter in each line is the AFP code page global identifier that maps to a Code Page Map file. (See “Code Page Map files” on page 315 for more information about mapping code pages.) The second parameter is the Windows character set that you decide is the best match for your AFP code page. The last line gives the default parameter values to be used when a default is required.


```
[CODEPG]
;codepage = cpgid,wincp
T1DCDCFS=1003,ANSI
T1DEBASE=2058,ANSI
T1D0BASE=2063,ANSI
T1D0GP12=2085,ANSI
T1GI0395=2079,ANSI
T1GPI363=2066,SYMBOL
T1V10037=37,ANSI
T1V10273=273,ANSI
T1000290=290,ANSI
T1000310=310,ANSI
T1000423=423,ANSI
T1000905=905,ANSI
DEFAULT =361,ANSI
```

Figure 8. Example of the Code Page Definition file (CPDEF.FNT) contents

Attribute	Possible Values	Shipped Default
Code Page Global Identifier	IBM-defined CPGID or your own defined CPGID within this range: 65280 to 65534	361
Windows Character Set	ANSI or SYMBOL	ANSI

Table 10. Code Page Definition file attribute values

Code Page Definition file rules

- Parameters must be separated by a comma. Table 10 lists the possible values and shipped default values for each parameter.
- Only the first parameter (code page identifier) is required.
- If you create your own code page, you must assign it a unique code page identifier. Leading zeros are not valid. (You may use an IBM code page global identifier but only if the character-to-hexadecimal code mapping is the same for your code page.)
- You can set a default code page. The default code page that is set within the file must be the last entry in the file.

Code Page Map files

IBM provides one Code Page Map file for each AFP code page supplied with Infoprint and the Data1 and Sonoran licensed programs. These files are installed in their own subdirectory (MAPS) under the FONT subdirectory. The file is named for its code page global identifier (CPGID) and has a file extension of .cp (for example, 2063.cp is the file name for the T1D0BASE code page map; its CPGID is 2063). Each file contains the character identifiers

(and associated EBCDIC hexadecimal code points) for an IBM code page and maps them to character identifiers (and associated ASCII code points) for a Windows ANSI or SYMBOL character set.

```
;T1000395 to ANSI
SP010000 40 SP010000 20
LA150000 42 LA150000 E2
LA170000 43 LA170000 E4
LA130000 44 LA130000 E0
SP180000 8B SP180000 BB
SM560000 8C SM560000 89
SA000000 8D SP100000 2D
LI510000 8E NOMATCH 00
LF570000 8F NOMATCH 00
SM190000 90 SM190000 B0
LJ010000 91 LJ010000 6A
LF510000 A0 NOMATCH 00
;;;;;;;;; ; SD150000 5E
;;;;;;;;; ; SD130000 60
;;;;;;;;; ; LT630000 FE
/*
```

Figure 9. Code Page Map file. Example of the partial contents of the Code Page Map file 395.cp for the T1000395 code page mapped to the Windows ANSI character set.

Code Page Map file rules

- Parameters must be separated by blanks.
- “NOMATCH” means there is not a matching character in the Windows character set.
- The “NOMATCH” hexadecimal code of 00 is mapped to the undefined code point. When a document contains a character that does not exist in the Windows character set, that character cannot be displayed on the screen. If the character has not been remapped in the Code Page Map file or the Alias file⁶, the undefined code point character will be displayed as a substitute. The character to be displayed for an undefined code point can be specified on the **Preferences** dialog box.
- The string of semi-colons (;;;;;;;;;;) means this line is ignored as a comment. It also indicates the Windows code page contains a character that does not exist in the IBM code page. The code point for a Windows character not found in the IBM code page can be used for replacing NOMATCH characters.

⁶ See “Alias file” on page 318 for more information about remapping code points.

Code Page Map file REXX program for building a Code Page Map file

IBM supplies a sample Restructured Extended Executor Language (REXX) program (BLDCPMAP.REX) that you can use to create Code Page Map files. This program executes in MVS, OS/2, OS/390, and Windows REXX environments. The REXX program is in the SAMPLES subdirectory of the FONT subdirectory.⁷

The BLDCPMAP.REX program requires a host AFP code page and one of the Windows character set files: ANSI.WCP or SYMBOL.WCP.⁸ The program's output is a Code Page Map file that maps the characters in the host code page to matching characters in the Windows character set so that it can be used with the client. It also identifies how many unmatched characters there are in the code page so that you can determine which Windows character set (ANSI or SYMBOL) contains the most matching characters. Matching is done using graphic character identifiers.

If you are going to use the BLDCPMAP.REX EXEC on your MVS or OS/390 system, make sure that you translate it from ASCII to EBCDIC and carriage return and line feed (CR/LF) must indicate a new line (see the BLDCPMAP.REX file prologue for details).

Setting up to build a Code Page Map file

You can either transfer the BLDCPMAP REXX program and the Windows character set file to your host system and run the program there, or you can transfer your AFP code pages to your workstation and run the program under Windows (if you have REXX installed on your workstation). You can use any file transfer program that handles standard host record format files and ASCII CR/LF line endings, with or without ASCII to EBCDIC translation. (We recommend that you use the IBM eNetwork Personal Communications program.)

If you transfer the REXX program and the Windows character set files (these files have an extension of .WCP) to your MVS or OS/390 host system, they must be translated from ASCII to EBCDIC and CR/LF must indicate a new line. All of the files transferred to the host system must be human-readable. If you transfer your AFP code pages to your Windows workstation, you *must* specify a **binary** format. If the file transfer is not correct, a REXX error occurs when you run the BLDCPMAP program.

On your Windows workstation, rename the BLDCPMAP.REX file to BLDCPMAP.CMD and ensure that REXX is installed. On your MVS or OS/390 system, the program may be run explicitly with the EXEC command or implicitly by member name, if the partitioned data set containing the BLDCPMAP program was previously allocated to your system file that contains

⁷ The FONT subdirectory is in the directory in which you installed the client.

⁸ The Windows character set files are shipped with the client and can be found in the SAMPLES subdirectory of the FONT subdirectory.

execs (usually SYSEXEC or SYSPROC). If the REXX program is named correctly, you can run the program without parameters to get the correct syntax of the command. You can also see the prologue for the EXEC for syntax.

When you run the BLDCPMAP program, and you have selected which Code Page Map file that you want to use with the client, place that Code Page Map file in the MAPS subdirectory of the FONT subdirectory in the directory in which you installed the client. Update the CPDEF.FNT file in the FONT subdirectory. In order for the client to find the Code Page Map file, it must be named as follows:

`code-page-global-identifier.CP`

For more information about using the Code Page Map file, see “Code Page Map files” on page 315. For more information about the BLDCPMAP program (for example, the syntax for running the program), see the prologue in the BLDCPMAP.REX file.

Alias file

The Alias file contains 2 sections: one section for font family name aliases [FONT] and one section for character identifier aliases [CHARID].

The first section, identified by the section header [FONT], lists the font familyname aliases. Font familyname aliases allow you to change all of the requested instances of a font familyname (as defined in the Character Set Definition file) to another font familyname. For example, this file is used to change all requests for the SonoranSerif font (which may not exist on the workstation) to requests for the TimesNewRoman font (which is one of the core fonts shipped with the client) as shown in Figure 10 on page 319.

ATM is the supported font program, however, TrueType fonts can be used with the client, but fidelity and character mapping will likely be incorrect. As a backup, a second font (TrueType) can be specified after the Type 1 font name. If the Type 1 font is not found, the TrueType font will be used to display your document.

Note: Be aware that font familyname remapping, especially to TrueType fonts, can cause some misalignment of text characters since the display font is not the same as the font used to create the AFP document. The font familyname can be found listed in the ATM Control Panel. Remapping of one font familyname to a different font familyname with very different characteristics (such as STYLE) may mean a matching font cannot be found. You will receive an error message if either font substitute cannot be found.

```
[FONT]
; ***** Requested font = Type 1 font, TrueType font *****
Book=TimesNewRoman,Times New Roman
CourierOverstrike=Courier,Courier New
SonoranSerif=TimesNewRoman,Times New Roman
SonoranSansSerif=Helvetica,Arial
Text=Courier,Courier New
```

Figure 10. The [FONT] section. This example of the [FONT] section is from the Alias file (ALIAS.FNT).

The second section, identified by the section header [CHARID], lists the character identifier aliases. Character identifier aliases (also known as glyph identifiers) allow you to change all of the requested instances of a character to another character. For example, since the Windows ANSI character set does not contain the ff ligature (LF510000), it is not mapped to a character in the code page map files (see Figure 9 on page 316). Instead, it is mapped to NOMATCH 00. If you want to map all occurrences of LF510000 — NOMATCH pair to a lower case f, you could do this in the [CHARID] section of the ALIAS.FNT file with the following entry:

```
LF510000=LF010000
```

If you want to change one specific character for one specific code page, then you may remap the character on that code page to another character as shown in Figure 9 on page 316.

The Alias file is checked only when a NOMATCH 00 is found in a character mapping.

Note: Using the Alias file for more than a few character substitutions is not recommended as program performance will be affected. If a lot of character substitutions are needed, it is better to make those changes directly to the mappings in the Code Page Map files that you are using.

```
[CHARID]
LF510000=LF010000
SA000000=SP320000,SP100000
```

Figure 11. The [CHARID] section. This example of the [CHARID] section is from the Alias file (ALIAS.FNT).

Alias File Rules

- For family name aliases, all requests for the first family name in the Character Set Definition file have the second family name substituted for them. If the second family name is not found, the TrueType font (the third family name) is requested.
- Only 2 family name substitutes per line are allowed (to the right of the equal sign), and they must be separated by a comma.

- If multiple mappings are listed in the file for the same family name, only the first match is used.
- The Alias file is processed sequentially and is *not* chained (for example, if “Century Schoolbook” is set equal to “Times,” and “Times” is set equal to “TimesNewRoman,” “Century Schoolbook” will *not* be set to “TimesNewRoman”).
- Blanks in family names are treated as characters (for example, “Times New Roman” is not the same font as “TimesNewRoman”).
- The [CHARID] section of the Alias file is only used if the second character identifier is NOMATCH 00.
- The character identifier that you want modified in the [CHARID] section must be followed by an equal sign and the character identifier to which it is to be changed. A character remap occurs when the modified character identifier (the character to the left of the equal sign in the [CHARID] section) matches the first character identifier of a non-matching pair in the Code Page Map file.
- Several character identifiers (substitute char id) may be listed to the right of the equal sign separated by commas. The first substitute character identifier is substituted for the modified character identifier unless it does not exist in the Windows font. If it does not exist, then the next substitute character identifier is used. If none of the substitute character identifiers exist, the undefined code point is used. If you want to see the contents of the Windows character sets, see the .WCP files in the SAMPLES subdirectory of the FONT directory.
- A maximum of 4 substitute character identifiers are allowed.

Support for TrueType fonts

The client supports Type 1 fonts installed under Adobe Type Manager (ATM). Type 1 outline fonts are supplied with the client and provide better fidelity in general than mapping to TrueType fonts. These fonts are installed in the directory specified in ATM or the FONTS subdirectory in the directory in which you installed the client. The Type 1 outline fonts that are installed with the client and their PC file names are as follows:

FONT NAME	PC FILE NAME
TIMESNEWROMAN	tnr*.*
HELVETICA	helv*.*
COURIER	cou*.*
BOLDFACE	bfc*.*
COURIERAPL2	apl*.*
GOTHICTEXT	got*.*
LETTERGOTHIC	lgo*.*
OCR_A	ocr_a.*
OCR_B	ocr_b.*
PRÉSTIGE	prs*.*

TrueType Fonts

When the client starts, it checks to see if ATM is available. If ATM is not available or the Type 1 font that you requested cannot be found, then the client uses TrueType fonts. If you do not install and use ATM, then you may use TrueType fonts to display your documents. To use TrueType fonts, you must do the following:

- If ATM is installed, it must be disabled, removed, or the Type 1 fonts installed with the client must be removed (if they are not used by other applications on your workstation). You can use ATM to remove these fonts.

To request a specific TrueType font, use the second font substitution family name in the ALIAS.FNT file as described in “Alias file” on page 318.

TrueType Font Substitution Problems: Make sure that the TrueType font that you have requested is installed on your workstation. Font substitutions that occur when fonts are not available may cause unexpected results when displaying your files. For example, Courier New is requested in the ALIAS.FNT file and is available with Windows NT, but is not available on Windows 2000; the same document may appear (view) differently on the two systems (however, the font can be installed on the Windows 2000 system).

Appendix A. Microsoft Visual Basic 5.0 DDE Program Sample

This program sample is provided on an as-is basis. A licensee of the OnDemand product is free to copy, revise, modify, and make derivative works of this program sample as they see fit.

This program was written and compiled using Microsoft Visual Basic 5.0 and demonstrates the use of the following OnDemand DDE Commands:

- ACTIVATE_DOC
- CLOSE_ALL_DOCS
- ENABLE_SWITCH
- EXIT
- GET_DOC_VALUES
- GET_NUM_DOCS_IN_LIST
- LOGON
- OPEN_DOC
- OPEN_FOLDER
- SEARCH_FOLDER
- SET_FIELD_DATA
- SET_FOCUS
- SHOW_WINDOW

Global Variables Used by the Demo

This section defines the global variables used by the Demo.

```
Option Explicit
Global Const apptitle = "OnDemand VB Demo"
Global Const yes = 1
Global Const no = 0
Global Const leave = 100000#

Global Const arstopic = "ARS|ARS"      'Default DDE application at topic

Global Const defini = "arsvblan.ini"   'Default ini file name
Global Const defstanza = "VBDEMO"     'Default stanza
Global Const arsgui = "ARSGUI32.EXE"  'Default Client exe

'Default Client parms
' Default Client startup parms
' /I   = enable DDE interface
' /B   = disable user confirmation
' /W N = invisible window (don't use during debugging)
' /K   = disable Exit (don't use during debugging)
' /V   = disable anticipation
Global Const arsguiopts = " /I /B /V /W N /K"

Global ininame As String               'Ini file name
Global server As String                'Server name
Global userid As String                'userid
Global pass As String                  'password
Global folder As String                'folder

Global guipath As String                'Client exe path
Global Const linktype = 2              'DDE linkage = manual
Global Const linktime = 3000           'DDE wait time

Global doc_ids(0 To 2) As String       'Doc ids returned on OPEN_DOC

Global txtStack(1 To 10) As String

'Define the Windows APIs used by the program
Declare Function GetModuleHandle Lib "Kernel" (ByVal lpModuleName As String) As Integer
Declare Function GetPrivateProfileString Lib "Kernel" (ByVal sname$, ByVal Kname$,
    ByVal Def$, ByVal ret$,ByVal Size%, ByVal FName$) As Integer
Declare Function SetFocusAPI Lib "User" Alias "SetFocus" (ByVal hWnd As Integer) As Integer
```

Entry Point for the Demo

The demo will drive the OnDemand Windows client via the DDE interface. Some important things to note: the commands sent to the OnDemand client are not DDE EXECUTE's, they are all DDE REQUEST's. This is important because sending DDE EXECUTE's to the OnDemand client will result in an error. Also it is important to start the OnDemand client with at least the /I option. This enables the DDE interface of the OnDemand client. (See the globals section to see what options were used for the VBDEMO.

The demo code was written using Visual Basic 5.0 (32-bit). The txtDemo control (which is hidden) is used as the DDE client in the DDE conversation with the OnDemand client window. In fncDDElink(), you will see where we setup this control to perform the DDE request and that the data comes back to this control. Therefore any data returned by the OnDemand client window will have to be parsed from out of this control.

```
Sub Main()  
    Dim rc As Integer  
  
    'Initialize globals and read in data from ini file(s).  
    Call fncInit  
  
    frmStatusDlg.lblStatus.Caption = "Starting client..."  
    frmStatusDlg.Show 0  
  
    'Start OnDemand client.  
    Shell (guipath + arsgui + arsguiopts)  
  
    'Logon to the server (logon information was gathered from ini  
    'file during fncInit. User cannot do anything else while this  
    'is going on. Try and use SetFocusAPI() to restore focus to our  
    'status message while this is going on.  
    frmStatusDlg.lblStatus.Caption = "Logging on to Server..."  
    Call frmCreditV1.fncLogon  
    rc = SetFocusAPI(frmStatusDlg.hWnd)  
  
    'Open the "Baxter Bay Credit" folder  
    frmStatusDlg.lblStatus.Caption = "Opening folder..."  
    Call frmCreditV1.fncOpenFolder  
  
    'Don't need the status message box any more.  
    frmStatusDlg.Hide  
  
    'Only after we have logged on and opened the folder do we  
    'display the VBDEMO form.  
    frmCreditV1.Show 1  
  
End Sub
```

```

'Send DDE REQUEST of CLOSE_ALL_DOCS to the client window:
Private Sub fncCloseDoc()
    Dim cmdline, qrc As String

    Call fncDispStatus("Close all open docs...")
    cmdline = "CLOSE_ALL_DOCS"
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    qrc = fncGetrc(txtDemo)
    If qrc <> "0" Then
        Call quit(cmdline, qrc)
    End If

    Call fncDispStatus("All open docs closed.")
End Sub

'This procedure handles the link to the OnDemand client.
' Topic should come is as ARS|ARS, this is the app name and topic name.
Private Sub fncDDElink(ByVal topic As String, ByVal cmd As String,
    ByVal mode As Integer, ByVal waittime As Integer)

'Setup local variables
    Dim sync, lntxtDemo, i, rc As Integer
    Dim workchar, workline, msg As String

'Set up error handler to show contact errors
    On Error GoTo HandleError

'Set up DDE link and pass required data:
    txtDemo = "-"
    txtDemo.LinkTimeout = waittime
    txtDemo.LinkTopic = topic
    txtDemo.LinkItem = cmd
    txtDemo.LinkMode = mode
    'Calling LinkRequest performs the request.
    txtDemo.LinkRequest
Exit Sub

```

```

HandleError:
    'Handle DDE errors
    rc = Err
    Select Case rc
        Case 280 To 297
            Select Case rc
                Case 280
                    msg = "DDE channel not closed; awaiting response from foreign application"
                Case 281
                    msg = "No more DDE channels"
                Case 282
                    msg = "DDE requests are being refused"
                Case 283
                    msg = "Too many apps responded"
                Case 284
                    msg = "DDE channel locked"
                Case 285
                    msg = "App is not accepting DDE requests..."
            End Select
        Case Else
            msg = "Non-DDE error occurred " + Str(rc)
        End Select
    MsgBox msg
    Resume Next
End Sub

```

```

'Used to send DDE REQUEST command of ACTIVATE_DOC or OPEN_DOC to
' the OnDemand client.
Private Sub fncDispDoc(ByVal docnum As Integer)
    Dim cmdline, qrc As String

    'If the document the user is requesting to be displayed has
    'previously been opened, then use ACTIVATE_DOC to redisplay
    'it, otherwise we will need to OPEN_DOC and store away the
    'document id.
    'If the user closes the document view from the client interface
    'we need to be notified of this event so that we can update
    'our doc_id array. Currently the client does not support this.
    If doc_ids(docnum) <> "0" Then
        Call fncDispStatus("Activating the document...")
        cmdline = "ACTIVATE_DOC /D " + doc_ids(docnum)
        Call fncDDElink(arstopic, cmdline, linktype, 3000)
        qrc = fncGetrc(txtDemo)
        If qrc <> "0" Then
            'The user possibly closed the view from the client,
            'reset the document id to 0 and tell the user to try again.
            doc_ids(docnum) = "0"
            Call fncDispStatus("Activating the document...ERROR")
            MsgBox "Could not activate, try to view again!"
            Exit Sub
        End If
        Call fncDispStatus("Activating the document...done")
    Else
        'Open the document
        Call fncDispStatus("Open the document...")
        cmdline = "OPEN_DOC /N " + Str(docnum)
        Call fncDDElink(arstopic, cmdline, linktype, 3000)
        qrc = fncGetrc(txtDemo)
        If qrc <> "0" Then
            Call quit(cmdline, qrc)
        End If
        doc_ids(docnum) = fncGetdochandle(txtDemo)
        Call fncDispStatus("Open the document...done.")
    End If

    'Make the display visible
    Call fncDispStatus("Opening the display...")
    cmdline = "SHOW_WINDOW /W"
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    qrc = fncGetrc(txtDemo)
    If qrc <> "0" Then
        Call quit(cmdline, qrc)
    End If
    Call fncDispStatus("Opening the display...done.")

    Call fncDispStatus("Document retrieval complete.")
End Sub

```

```

'Obtains the hitlist of documents from the OnDemand client.
Private Sub fncGetHitlist()
    Dim cmdline, qrc As String
    Dim num_docs As Integer

    'Get the number of documents which matched the search
    'criteria.
    cmdline = "GET_NUM_DOCS_IN_LIST"
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    num_docs = CInt(Mid(txtDemo, 3, 10))
    If num_docs = 0 Then
        MsgBox "No documents found matching search criteria!"
        Exit Sub
    End If

    Call fncDispStatus("Getting account information...")

    'Get the first document and parse its data to display
    cmdline = "GET_DOC_VALUES /N 0"
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    Call fncExtract(txtDemo.Text)
    'Display its data
    pnlPayData1.Caption = txtStack(1)
    Panel3D1.Caption = txtStack(4)
    'Add about 20 days or so to the statement date'
    Panel3D2.Caption = fncParseDate(txtStack(1))
    cmdViewStmt1.Enabled = True

    'If there are at lease two documents then get number 2
    If num_docs > 1 Then
        cmdline = "GET_DOC_VALUES /N 1"
        Call fncDDElink(arstopic, cmdline, linktype, 3000)
        Call fncExtract(txtDemo.Text)
        'Display its data
        pnlPayData2.Caption = txtStack(1)
        Panel3D3.Caption = txtStack(4)
        'Add about 20 days or so to the statement date'
        Panel3D4.Caption = fncParseDate(txtStack(1))
        cmdViewStmt2.Enabled = True
    Else
        'There was only 1 document so disable 2nd "View" button.
        cmdViewStmt2.Enabled = False
    End If

```

```

'If there are at lease three documents then get number 3
If num_docs > 2 Then
    cmdline = "GET_DOC_VALUES /N 2"
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    Call fncExtract(txtDemo.Text)
    'Display its data
    pnlPayData3.Caption = txtStack(1)
    Panel3D5.Caption = txtStack(4)
    'Add about 20 days or so to the statement date'
    Panel3D6.Caption = fncParseDate(txtStack(1))
    cmdViewStmt3.Enabled = True
Else
    'There were only 2 documents so disable 3rd "View" button.
    cmdViewStmt3.Enabled = False
End If
Call fncDispStatus("Getting account information...done.")
End Sub

```



```

'Procedure used to hide the OnDemand client window.
'Sends a DDE REQUEST message of SHOW_WINDOW to the client.
Private Sub fncHideWindow()
    Dim cmdline, qrc As String

    cmdline = "SHOW_WINDOW /W N"
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    qrc = fncGetrc(txtDemo)
    If qrc <> "0" Then
        Call quit(cmdline, qrc)
    End If
End Sub

'Logon to the OnDemand client.
Public Sub fncLogon()
    Dim cmdline, qrc As String

    Call fncDispStatus("Logon to Client...")
    cmdline = "LOGON /S " + server + " /U " + userid + " /P " + pass
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    qrc = fncGetrc(txtDemo)
    If qrc <> "0" Then
        'If we fail the logon the client will display his logon dialog.
        'We will not return from this DDE call until the user either
        'successfully log's onto a server or cancel's the process, in
        'which case we end up with an error code and inside of this If
        'statement. Close the client and then ourselves.
        'I am not sure if the above statement is valid if you started up
        ' the OnDemand client with the Disable anticipation (/V) parameter.
        Call fncDDElink(arstopic, "EXIT", linktype, 3000)
        Call fncDispStatus("Logon to client...failed.")
        End
    End If
    Call fncDispStatus("Logon to Client...done.")
End Sub

'Open up an OnDemand folder.
Public Sub fncOpenFolder()
    Dim cmdline, qrc As String

    Call fncDispStatus("Open the folder...")
    cmdline = "OPEN_FOLDER /F " + folder
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    qrc = fncGetrc(txtDemo)
    If qrc <> "0" Then Call quit(cmdline, qrc)
    Call fncDispStatus("Open the folder...done.")

End Sub

```

```

'Search the OnDemand folder for documents.
Private Sub fncSearchDoc(ByVal AcctNum As String)
    Dim cmdline, qrc As String

    'Setup our search fields with the client.
    Call fncDispStatus("Setting up Search...")
    cmdline = "SET_FIELD_DATA /F Account /1 " + AcctNum
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    qrc = fncGetrc(txtDemo)
    If qrc <> "0" Then
        Call quit(cmdline, qrc)
    End If
    Call fncDispStatus("Setting up Search...done.")

    'Have the client perform the search.
    Call fncDispStatus("Performing the Search...")
    cmdline = "SEARCH_FOLDER"
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    qrc = fncGetrc(txtDemo)
    If qrc <> "0" Then
        Call quit(cmdline, qrc)
    End If
    Call fncDispStatus("Performing the Search...done.")
End Sub

'Performs three DDE steps for us:
'   - Inform client to retrieve selected document.
'   - Enable the switch back toolbar button on the clients toolbar so that the
'     user can get back easily to the VBDEMO.
'   - Switch focus to the client.
Private Sub fncViewDoc(ByVal docnum As Integer)
    'Setup local variables
    Dim MyHandle As Integer

    'Display the document
    Call fncDispDoc(docnum)
    'Activate DDE and transfer Focus to OnDemand
    MyHandle = frmCreditV1.hWnd
    Call fncDDElink(arstopic, "ENABLE_SWITCH /H " + Str(MyHandle) + " /C " + apptitle,
    linktype, 3000)
    Call fncDDElink(arstopic, "SET_FOCUS", linktype, 3000)
End Sub

'Displays error code.
Private Sub quit(ByVal qinfo As String, ByVal qrc As String)
    Dim quitstring As String

    quitstring = "Error encountered: " + qinfo + " rc=" + qrc
    MsgBox quitstring
End
End Sub

```

```

'GUI control used to display customer information.
'We do not obtain the customer information from out of OnDemand, it is
' not stored there. The normal way to obtain this information would be
' to get it out of your business database. After which you would look up
' the customer statements in OnDemand. We simply get this information from
' out of an ini file.
Private Sub cmdCustInfo_Click()
    Dim acct_num, ini_str As String
    Dim cmdline, qrc As String
    Dim rc As Integer
    Dim first_num, second_num, third_num As Integer

    'Zero out the Payment record fields before retrieving new customer
    pnlPayData1.Caption = ""
    Panel3D1.Caption = ""
    Panel3D2.Caption = ""
    pnlPayData2.Caption = ""
    Panel3D3.Caption = ""
    Panel3D4.Caption = ""
    pnlPayData3.Caption = ""
    Panel3D5.Caption = ""
    Panel3D6.Caption = ""
    'Zero out the Customer Information fields.
    pnlNameData.Caption = ""
    pnlSSNData.Caption = ""
    pnlDOBData.Caption = ""
    pnlMNameData.Caption = ""
    pnlAddrData1.Caption = ""
    pnlAddrData2.Caption = ""
    pnlPhoneData.Caption = ""

    'Disable "View" buttons
    cmdViewStmt1.Enabled = False
    cmdViewStmt2.Enabled = False
    cmdViewStmt3.Enabled = False

    'Hide client window
    Call fncHideWindow

    'Look up the account number, contained in the pnlAcctnumData text field
    'in the arsvblan.ini file. If found, read the respective
    'fields. If not found display error message.
    acct_num = txtAcctnumData.Text

    'Do at least a little validation.
    If Len(acct_num) <> 11 Then
        MsgBox "Correct format for account # is 000-000-000"
        Exit Sub
    End If

```

```

'If we have gotten to here we know that we have an account
'number of the format 000-000-000. If either of the first
'two sections of the number are non-zero or if the third
'section is not between 001-046 then default to the account
'number 000-000-001.
first_num = Int(Mid(acct_num, 1, 3))
second_num = Int(Mid(acct_num, 5, 3))
third_num = Int(Mid(acct_num, 9, 3))
If first_num <> 0 Or second_num <> 0 Or third_num > 46 Then
    acct_num = "000-000-001"
ElseIf third_num = 0 Then
    MsgBox "Invalid account number!"
    Exit Sub
End If

ini_str = fncParmGet(acct_num, "Name", ininame)
If Len(ini_str) = 0 Then
    MsgBox "'Name' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pn1NameData.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "SSN", ininame)
If Len(ini_str) = 0 Then
    MsgBox "'SSN' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pn1SSNData.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "DOB", ininame)
If Len(ini_str) = 0 Then
    MsgBox "'DOB' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pn1DOBData.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "MaidenName", ininame)
If Len(ini_str) = 0 Then
    MsgBox "'MaidenName' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub

```

```

End If
pnlMNameData.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "StreetAddress", ininame)
If Len(ini_str) = 0 Then
    MsgBox "'StreetAddress' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlAddrData1.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "CityStateZip", ininame)
If Len(ini_str) = 0 Then
    MsgBox "'CityStateZip' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlAddrData2.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "PhoneNum", ininame)
If Len(ini_str) = 0 Then
    MsgBox "'PhoneNum' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlPhoneData.Caption = " " + ini_str

'We are changing customer accounts so before we get new customer
'information, close old customers open documents.
If doc_ids(0) <> "0" Or doc_ids(1) <> "0" Or doc_ids(2) <> "0" Then
    doc_ids(0) = "0"
    doc_ids(1) = "0"
    doc_ids(2) = "0"
    cmdline = "CLOSE_ALL_DOCS"
    Call fncDDElink(arstopic, cmdline, linktype, 3000)
    qrc = fncGetrc(txtDemo)
    If qrc <> "0" Then
        Call quit(cmdline, qrc)
    End If
End If

'Set up the search fields and perform search.
Call fncSearchDoc(acct_num)

'Get the 3 most recent statements.
Call fncGetHitlist

'Give ourselves back the focus
rc = SetFocusAPI(frmCreditV1.hWnd)
End Sub

```

'User has chosen to exit the VBDEMO. Before exiting close down the client.

```
Private Sub cmdExit_Click()
```

```
    Dim OnDemandHandle As Integer
```

```
    Call fncDispStatus("Program ending...")
```

```
    'Determine if OnDemand is loaded
```

```
    OnDemandHandle = GetModuleHandle(arsgui)
```

```
    'If not loaded, then quit, else shutdown
```

```
    If OnDemandHandle > 0 Then
```

```
        Call fncDispStatus("Shutting Client Down...")
```

```
        Call fncDDElink(arstopic, "EXIT", linktype, linktime)
```

```
    End If
```

```
    'Terminate the VBDEMO
```

```
    End
```

```
End Sub
```

'View button number 1. Have the client retrieve the first document in
' the hitlist and display it.

```
Private Sub cmdViewStmt1_Click()
```

```
    Call fncViewDoc(0)
```

```
End Sub
```

'View button number 2. Have the client retrieve the second document in
' the hitlist and display it.

```
Private Sub cmdViewStmt2_Click()
```

```
    Call fncViewDoc(1)
```

```
End Sub
```

'View button number 3. Have the client retrieve the third document in
' the hitlist and display it.

```
Private Sub cmdViewStmt3_Click()
```

```
    Call fncViewDoc(2)
```

```
End Sub
```

```

'If the user is not using the Exit button to close down the demo
' this function will be called as a result of the form being unloaded
' so go ahead and shut down the client then exit.

```

```

Private Sub Form_Unload(Cancel As Integer)
    Dim OnDemandHandle As Integer

    Call fncDispStatus("Program ending...")

    'Determine if OnDemand is loaded
    OnDemandHandle = GetModuleHandle(arsgui)
    'If not loaded, then quit, else shutdown
    If OnDemandHandle > 0 Then
        Call fncDispStatus("Shutting Client Down...")
        Call fncDDElink(arstopic, "EXIT", linktype, linktime)
    End If

```

```

    'Terminate the VBDEMO
End
End Sub

```

```

'Make sure that the data they are entering for the account number
' is valid.

```

```

Private Sub txtAcctnumData_KeyPress(KeyAscii As Integer)

```

```

    Dim pos As Integer

```

```

    pos = txtAcctnumData.SelStart

```

```

    Select Case KeyAscii
        Case 48 To 59
            'pos must be 0-2, 4-6, 8-10
            Select Case pos
                Case 0 To 2, 4 To 6, 8 To 10
                    'OK
                Case Else
                    Beep
                    KeyAscii = 0
            End Select
        Case 45 ' the - character
            'pos must be 3 or 7
            If pos <> 3 And pos <> 7 Then
                Beep
                KeyAscii = 0
            End If
        Case 8, 127
            'Just let these through.
        Case Else
            Beep
            KeyAscii = 0
    End Select

```

```

End Sub

```

'This procedure fills in the status line on the form and left adjusts.

```
Public Sub fncDispStatus(ByVal status As String)
    frmCreditV1.pnlStatus.Caption = status + Space$(255)
End Sub
```

'This procedure breaks out the words of the input

'string. Words must be delimited by a tab character.

'The words are stored in the global string array txtStack.

```
Sub fncExtract(ByVal workstring As String)
    Dim txtptr, lenstring, i As Integer
    Dim tabchar, workline, workchar As String

    txtptr = 0
    tabchar = Chr(9)
    workline = ""
    lenstring = Len(workstring)
    workstring = Mid$(workstring, 3, lenstring)

    'Extract chars to the first blank
    For i = 1 To lenstring
        workchar = Mid$(workstring, i, 1)
        'When a tab is found, store result, reset
        If workchar = tabchar Then
            txtptr = txtptr + 1
            txtStack(txtptr) = workline
            workline = ""
        'Otherwise, keep building the work string
        Else
            workline = workline + workchar
        End If
    Next

    If Len(workline) > 0 Then
        txtptr = txtptr + 1
        txtStack(txtptr) = workline
    End If
End Sub
```

'This function extracts out the document handle from the
'return string.

```
Function fncGetdochandle(ByVal workstring As String)
    Dim lenstring, first, i As Integer
    Dim rc, workline, workchar As String

    'Set the return code for invalid function call
    rc = "999"
    first = yes
    workstring = Trim$(workstring)
    lenstring = Len(workstring)
```



```

'Extract chars to the first blank
If lenstring > 0 Then
    workline = ""
    For i = 1 To lenstring
        workchar = Mid$(workstring, i, 1)
        'When a second blank is found, stop
        If workchar = " " Then
            If first = yes Then
                first = no
                workline = ""
            Else
                rc = workline
                i = leave
            End If
            'Otherwise build up return code
        Else
            workline = workline + workchar
        End If
    Next
    'If the doc handle has been built, assign it
    If workline <> "" Then
        rc = workline
    End If
End If

'Set the function return value
fncGetdochandle = rc

```

End Function

```

'This function extracts out the return code from the
'return string.
Function fncGetrc(ByVal workstring As String)
    Dim lenstring, i As Long
    Dim rc, workline, workchar As String

    'Set the return code for invalid function call
    rc = "999"
    workstring = Trim$(workstring)
    lenstring = Len(workstring)

    'Extract chars to the first blank
    If lenstring > 0 Then
        workline = ""
        For i = 1 To lenstring
            workchar = Mid$(workstring, i, 1)
            'When a blank is found, stop
            If workchar = " " Then
                rc = workline
                i = leave
            'Otherwise build up return code
            Else
                workline = workline + workchar
            End If
        Next
        'If a return code has been built, assign it
        If workline <> "" Then
            rc = workline
        End If
    End If

    'Set the function return value
    fncGetrc = rc

End Function

```

```

'Perform global initialization
Sub fncInit()
    Dim ini_str As String

    'Set document ids for all three to 0
    doc_ids(0) = "0"
    doc_ids(1) = "0"
    doc_ids(2) = "0"

    'Disable "View" buttons
    frmCreditV1.cmdViewStmt1.Enabled = False
    frmCreditV1.cmdViewStmt2.Enabled = False
    frmCreditV1.cmdViewStmt3.Enabled = False

    'The VBDEMO keyword in the PATHS stanza of the ars.ini file
    'points to the .ini file where the other
    'demo settings can be picked up. If the
    'VBDEMO keyword cannot be found, the ini
    'file is set to arsvblan.ini.

    'Try to find vbdemo inifile name
    ininame = defini
    ini_str = fncParmGet("PATHS", "VBDEMO", "ars.ini")
    'If the ini name is found, then set
    If Len(ini_str) > 0 Then
        ininame = ini_str
    End If

    'Try to find arsgui execution path
    ini_str = fncParmGet(defstanza, "GUIPath", ininame)
    'If it can't be found, check for an env var
    If Len(ini_str) = 0 Then
        MsgBox "Cannot find GUIPath in " + ininame
    End If

    'If the path is found, then set
    If Len(ini_str) > 0 Then
        guipath = ini_str + "\"
    End If

    'Try to find the server in the ars ini file
    ini_str = fncParmGet(defstanza, "Server", ininame)
    'If it can't be found, check for an env var
    If Len(ini_str) = 0 Then
        MsgBox "Cannot find Server in " + ininame
    End If
    If Len(ini_str) > 0 Then
        server = ini_str
    End If

```

```

'Try to find the userid in the ars ini file
ini_str = fncParmGet(defstanza, "Userid", ininame)
'If it can't be found, check for an env var
If Len(ini_str) = 0 Then
    MsgBox "Cannot find Userid in " + ininame
End If
If Len(ini_str) > 0 Then
    userid = ini_str
End If

'Try to find the password in the ars ini file
ini_str = fncParmGet(defstanza, "Password", ininame)
'If it can't be found, check for an env var
If Len(ini_str) = 0 Then
    MsgBox "Cannot find Password in " + ininame
End If
If Len(ini_str) > 0 Then
    pass = ini_str
End If
If pass = "<NULL>" Then
    pass = ""
End If

'Try to find the folder in the ars ini file
ini_str = fncParmGet(defstanza, "Folder", ininame)
folder = ini_str

End Sub

'This function returns information from the ini file.
Function fncParmGet(ByVal stanza As String, ByVal keyname As String, ByVal inifile As String)
    Dim Default, result As String
    Dim rc As Integer

    Default = ""
    result = Space$(255)

    rc = GetPrivateProfileString(stanza, keyname, Default, result, Len(result), inifile)
    If rc Then
        fncParmGet = Trim$(result)
        If Len(fncParmGet) > 1 Then
            fncParmGet = Left$(fncParmGet, Len(fncParmGet) - 1)
        End If
    Else
        fncParmGet = ""
    End If

End Function

```

```

'This function is only used to dummy up the date paid
'field of the form. The reason being is that for the
'demo, which uses the 'Baxter Bay Credit' folder,
'we cannot get this information from the database.
'This function adds approximately 20 days to the statement
'date field (which is passed in).
Public Function fncParseDate(ByVal stmtdate As String)
    Dim date_array(1 To 3) As String
    Dim searchch, workline, workchar As String
    Dim txtptr, lenstring, i As Integer
    Dim pay_day, pay_month, pay_year As Integer

    txtptr = 0
    searchch = Chr(47)
    workline = ""
    lenstring = Len(stmtdate)

    'Extract chars to the first '/'
    For i = 1 To lenstring
        workchar = Mid$(stmtdate, i, 1)
        'When a '/' is found, store result, reset
        If workchar = searchch Then
            txtptr = txtptr + 1
            date_array(txtptr) = workline
            workline = ""
            'Otherwise, keep building the work string
        Else
            workline = workline + workchar
        End If
    Next

    If Len(workline) > 0 Then
        txtptr = txtptr + 1
        date_array(txtptr) = workline
    End If

```

```

'date_array contains three elements, the first is the month
'number, the second is the day of the month and third is
'the year. Simply check if the day of the month plus 20
'is greater than 28, if so the difference becomes the new
'day of the month and we increment the month number.
pay_day = Int(date_array(2)) + 20
pay_month = Int(date_array(1))
pay_year = Int(date_array(3))
If pay_day > 28 Then
    pay_day = pay_day - 28
    pay_month = pay_month + 1
    If pay_month > 12 Then
        pay_month = 1
        pay_year = pay_year + 1
    End If
End If

fncParseDate = LTrim(Str(pay_month)) + "/" + LTrim(Str(pay_day)) + "/" + LTrim(Str(pay_year))
End Function

```

Appendix B. Microsoft VC++ 5.0 DDE Program Sample

This program sample is provided on an as-is basis. A licensee of the OnDemand product is free to copy, revise, modify, and make derivative works of this program sample as they see fit.

This program was written and compiled using Microsoft VC++ 5.0 and demonstrates the use of the following OnDemand DDE Commands:

```
CLOSE_DOC
CLOSE_FOLDER
ENABLE_SWITCH
EXIT
GET_DOC_VALUES
GET_NUM_DOCS_IN_LIST
GET_PRINTERS
LOGOFF
LOGON
OPEN_DOC
OPEN_FOLDER
PRINT_DOC
SEARCH_FOLDER
SHOW_WINDOW

#include "stdafx.h"
#include <ddeml.h>
#include <winpool.h>

#include "vcdde32.h"
#include "MainDlg.h"
#include "arsddeex.h"           // Shipped with OnDemand

static CMainDlg * pMainDlg;
static char      RequestedData[10000];           // Returned data from DDE command
static HSZ      hsz1, hsz2;
static DWORD    DdeInstance;
static HCONV    hDdeConv;

extern CDdeTestApp * pApp;           // Pointer to application instance

#define ERROR_MAP struct _ErrorMap
ERROR_MAP
{
    int    code;
    char * pMsg;
};
```

```

static ERROR_MAP Errors[] =
{ { ARS_DDE_RC_UNKNOWN_COMMAND, "Unknown command." },
  { ARS_DDE_RC_PARAM_NOT_SPECIFIED, "Parameter not specified." },
  { ARS_DDE_RC_INVALID_PARAM_VALUE, "Invalid parameter value." },
  { ARS_DDE_RC_SERVER_ERROR, "Server error." },
  { ARS_DDE_RC_FILE_ERROR, "File error." },
  { ARS_DDE_RC_NOT_LOGGED_ON, "Not logged on." },
  { ARS_DDE_RC_MAX_FOLDERS_OPEN, "Maximum folders open." },
  { ARS_DDE_RC_FOLDER_NOT_OPEN, "Folder not open." },
  { ARS_DDE_RC_NO_DOC, "No document exists." },
  { ARS_DDE_RC_NO_ACTIVE_DOC, "No document is active." },
  { ARS_DDE_RC_USER_ACTION_IN_PROGRESS, "User action in process." },
  { ARS_DDE_RC_UNAUTHORIZED_OPERATION, "Unauthorized operation." },
  { ARS_DDE_RC_USER_CANCELLED_OPERATION, "User cancelled operation." },
  { ARS_DDE_RC_INVALID_APPL_GROUP_NAME, "Invalid Appl Group Name." },
  { ARS_DDE_RC_INVALID_APPL_NAME, "Invalid Appl Name." },
  { ARS_DDE_RC_INVALID_INTEGER_FIELD, "Invalid integer field." },
  { ARS_DDE_RC_INVALID_DECIMAL_FIELD, "Invalid decimal field." },
  { ARS_DDE_RC_INVALID_DATE_FIELD, "Invalid date field." },
  { ARS_DDE_RC_INVALID_APPLGRP_FIELD_TYPE, "Invalid Appl Group field type." } };

#define NUM_ERRORS ( sizeof(Errors) / sizeof(ERROR_MAP) )

#define ADV_MAP struct _AdvMap
ADV_MAP
{
    char * pAdvData;
    char * pMsg;
};

static ADV_MAP Advises[] =
{ { ARS_DDE_EVENT_CRITERIA_BUTTON_1, "DDE Criteria 1 Button Clicked." },
  { ARS_DDE_EVENT_CRITERIA_BUTTON_2, "DDE Criteria 2 Button Clicked." },
  { ARS_DDE_EVENT_CRITERIA_BUTTON_3, "DDE Criteria 3 Button Clicked." },
  { ARS_DDE_EVENT_CRITERIA_BUTTON_4, "DDE Criteria 4 Button Clicked." },
  { ARS_DDE_EVENT_CRITERIA_BUTTON_5, "DDE Criteria 5 Button Clicked." },
  { ARS_DDE_EVENT_DOCLIST_BUTTON_1, "DDE Doclist 1 Button Clicked." },
  { ARS_DDE_EVENT_DOCLIST_BUTTON_2, "DDE Doclist 2 Button Clicked." },
  { ARS_DDE_EVENT_DOCLIST_BUTTON_3, "DDE Doclist 3 Button Clicked." },
  { ARS_DDE_EVENT_DOCLIST_BUTTON_4, "DDE Doclist 4 Button Clicked." },
  { ARS_DDE_EVENT_DOCLIST_BUTTON_5, "DDE Doclist 5 Button Clicked." },
  { ARS_DDE_EVENT_SWITCH_FOCUS, "Switch focus requested." },
  { ARS_DDE_EVENT_SWITCH_FOCUS_2, "Switch focus *** 2 *** requested." },
  { ARS_DDE_EVENT_SWITCH_FOCUS_3, "Switch focus *** 3 *** requested." },
  { ARS_DDE_EVENT_SWITCH_FOCUS_4, "Switch focus *** 4 *** requested." },
  { ARS_DDE_EVENT_SWITCH_FOCUS_5, "Switch focus *** 5 *** requested." } };

```



```

#define NUM_ADVISES ( sizeof(Advices) / sizeof(ADV_MAP) )

// DDE variables and functions

static HDEDDATA hDdeData, hDdeResult;

HDEDDATA FAR PASCAL DdeCallBack ( UINT      iType,
                                   UINT      iFmt,
                                   HCONV     hConv,
                                   HSZ       hsz1,
                                   HSZ       hsz2,
                                   HDEDDATA  hData,
                                   DWORD      dwData1,
                                   DWORD      dwData2 )
{
    int      j;
    char * pData;
    DWORD    data_len;

    switch ( iType )
    {
        case XTYP_DISCONNECT:
            hDdeConv = NULL;
            break;
        case XTYP_ADVDATA:
            if ( hData == NULL )
                AfxMessageBox( "hData is NULL in XTYP_ADVDATA" );
            else
            {
                pData = (char*)DdeAccessData( hData, &data_len );
                for ( j = 0; j < NUM_ADVISES; j++ )
                    if ( strcmp( Advices[j].pAdvData, pData ) == 0 )
                        break;
                AfxMessageBox( j < NUM_ADVISES
                               ? Advices[j].pMsg
                               : "Logic Error - invalid ADVDATA." );
                DdeUnaccessData( hData );
            }
            break;
    }
    return NULL;
}

```

```

static BOOL DoDdeCommand( char * pCommand, char * pParms )
{
    DWORD    data_len;
    char * pString1, * pData, * pFirstChar;
    int      j, rc;

    if ( pParms == NULL )
        pParms = "";
    pString1 = new char[ strlen( pCommand ) + strlen( pParms ) + 2 ];
    strcpy( pString1, pCommand );
    strcat( pString1, " " );
    strcat( pString1, pParms );

    hsz1 = DdeCreateStringHandle( DdeInstance, pString1, 0 );
    hDdeResult = DdeClientTransaction( NULL,
                                       0,
                                       hDdeConv,
                                       hsz1,
                                       CF_TEXT,
                                       XTYP_REQUEST,
                                       120000L,
                                       NULL );
    DdeFreeStringHandle( DdeInstance, hsz1 );

    delete pString1;

    RequestedData[0] = '\0';
    if ( hDdeResult == NULL )
    {
        int      error;
        char * pErr;
    }
}

```

```

error = DdeGetLastError( DdeInstance );
switch ( error )
{
    case DMLERR_ADVACKTIMEOUT:
        pErr = "DdeClientTransaction failed with DMLERR_ADVACKTIMEOUT";
        break;
    case DMLERR_BUSY:
        pErr = "DdeClientTransaction failed with DMLERR_BUSY";
        break;
    case DMLERR_DATAACKTIMEOUT:
        pErr = "DdeClientTransaction failed with DMLERR_DATAACKTIMEOUT";
        break;
    case DMLERR_DLL_NOT_INITIALIZED:
        pErr = "DdeClientTransaction failed with DMLERR_DLL_NOT_INITIALIZED";
        break;
    case DMLERR_DLL_USAGE:
        pErr = "DdeClientTransaction failed with DMLERR_DLL_USAGE";
        break;
    case DMLERR_EXECACKTIMEOUT:
        pErr = "DdeClientTransaction failed with DMLERR_EXECACKTIMEOUT";
        break;
    case DMLERR_INVALIDPARAMETER:
        pErr = "DdeClientTransaction failed with DMLERR_INVALIDPARAMETER";
        break;
    case DMLERR_LOW_MEMORY:
        pErr = "DdeClientTransaction failed with DMLERR_LOW_MEMORY";
        break;
    case DMLERR_MEMORY_ERROR:
        pErr = "DdeClientTransaction failed with DMLERR_MEMORY_ERROR";
        break;
    case DMLERR_NO_CONV_ESTABLISHED:
        pErr = "DdeClientTransaction failed with DMLERR_NO_CONV_ESTABLISHED";
        break;
    case DMLERR_NOTPROCESSED:
        pErr = "DdeClientTransaction failed with DMLERR_NOTPROCESSED";
        break;
    case DMLERR_POKEACKTIMEOUT:
        pErr = "DdeClientTransaction failed with DMLERR_POKEACKTIMEOUT";
        break;
}

```

```

        case DMLERR_POSTMSG_FAILED:
            pErr = "DdeClientTransaction failed with DMLERR_POSTMSG_FAILED";
            break;
        case DMLERR_REENTRANCY:
            pErr = "DdeClientTransaction failed with DMLERR_REENTRANCY";
            break;
        case DMLERR_SERVER_DIED:
            pErr = "DdeClientTransaction failed with DMLERR_SERVER_DIED";
            break;
        case DMLERR_SYS_ERROR:
            pErr = "DdeClientTransaction failed with DMLERR_SYS_ERROR";
            break;
        case DMLERR_UNADVACKTIMEOUT:
            pErr = "DdeClientTransaction failed with DMLERR_UNADVACKTIMEOUT";
            break;
        case DMLERR_UNFOUND_QUEUE_ID:
            pErr = "DdeClientTransaction failed with DMLERR_UNFOUND_QUEUE_ID";
            break;
    }
    AfxMessageBox( pErr );
    return FALSE;
}
else
{
    pData = (char*)DdeAccessData( hDdeResult, &data_len );
    rc = atoi( pData );
    if ( rc == ARS_DDE_RC_NO_ERROR )
    {
        pFirstChar = strchr( pData, ' ' );
        strcpy( RequestedData, &pFirstChar[1] );
    }
    else
    {
        for ( j = 0; j < NUM_ERRORS; j++ )
            if ( Errors[j].code == rc )
                break;
        AfxMessageBox( j < NUM_ERRORS
                        ? Errors[j].pMsg
                        : "Logic Error - invalid return code." );
    }
    DdeUnaccessData( hDdeResult );
    return rc == ARS_DDE_RC_NO_ERROR;
}
}

```

```

static BOOL DoAdviseLoop( char * pName, BOOL stop )
{
    hsz1 = DdeCreateStringHandle( DdeInstance, pName, 0 );
    hDdeResult = DdeClientTransaction( NULL,
                                       0,
                                       hDdeConv,
                                       hsz1,
                                       CF_TEXT,
                                       stop ? XTYP_ADVSTOP : XTYP_ADVSTART,
                                       120000L,
                                       NULL );
    DdeFreeStringHandle( DdeInstance, hsz1 );

    if ( hDdeResult == NULL )
    {
        int    error;
        char * pErr;
    }
}

```

```

error = DdeGetLastError( DdeInstance );
switch ( error )
{
    case DMLERR_ADVACKTIMEOUT:
        pErr = "DdeClientTransaction failed with DMLERR_ADVACKTIMEOUT";
        break;
    case DMLERR_BUSY:
        pErr = "DdeClientTransaction failed with DMLERR_BUSY";
        break;
    case DMLERR_DATAACKTIMEOUT:
        pErr = "DdeClientTransaction failed with DMLERR_DATAACKTIMEOUT";
        break;
    case DMLERR_DLL_NOT_INITIALIZED:
        pErr = "DdeClientTransaction failed with DMLERR_DLL_NOT_INITIALIZED";
        break;
    case DMLERR_DLL_USAGE:
        pErr = "DdeClientTransaction failed with DMLERR_DLL_USAGE";
        break;
    case DMLERR_EXEACKTIMEOUT:
        pErr = "DdeClientTransaction failed with DMLERR_EXEACKTIMEOUT";
        break;
    case DMLERR_INVALIDPARAMETER:
        pErr = "DdeClientTransaction failed with DMLERR_INVALIDPARAMETER";
        break;
    case DMLERR_LOW_MEMORY:
        pErr = "DdeClientTransaction failed with DMLERR_LOW_MEMORY";
        break;
    case DMLERR_MEMORY_ERROR:
        pErr = "DdeClientTransaction failed with DMLERR_MEMORY_ERROR";
        break;
    case DMLERR_NO_CONV_ESTABLISHED:
        pErr = "DdeClientTransaction failed with DMLERR_NO_CONV_ESTABLISHED";
        break;
    case DMLERR_NOTPROCESSED:
        pErr = "DdeClientTransaction failed with DMLERR_NOTPROCESSED";
        break;
    case DMLERR_POKEACKTIMEOUT:
        pErr = "DdeClientTransaction failed with DMLERR_POKEACKTIMEOUT";
        break;
    case DMLERR_POSTMSG_FAILED:
        pErr = "DdeClientTransaction failed with DMLERR_POSTMSG_FAILED";
        break;
}

```

```

        case DMLERR_REENTRANCY:
            pErr = "DdeClientTransaction failed with DMLERR_REENTRANCY";
            break;
        case DMLERR_SERVER_DIED:
            pErr = "DdeClientTransaction failed with DMLERR_SERVER_DIED";
            break;
        case DMLERR_SYS_ERROR:
            pErr = "DdeClientTransaction failed with DMLERR_SYS_ERROR";
            break;
        case DMLERR_UNADVACKTIMEOUT:
            pErr = "DdeClientTransaction failed with DMLERR_UNADVACKTIMEOUT";
            break;
        case DMLERR_UNFOUND_QUEUE_ID:
            pErr = "DdeClientTransaction failed with DMLERR_UNFOUND_QUEUE_ID";
            break;
    }
    AfxMessageBox( pErr );
    return FALSE;
}
else
    return TRUE;
}

////////////////////////////////////
// CMainDlg dialog

CMainDlg::CMainDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CMainDlg::IDD, pParent)
{
   //{{AFX_DATA_INIT(CMainDlg)
    // NOTE: the ClassWizard will add member initialization here
    //}}AFX_DATA_INIT
    // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CMainDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CMainDlg)
    // NOTE: the ClassWizard will add DDX and DDV calls here
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CMainDlg, CDialog)
    //{{AFX_MSG_MAP(CMainDlg)
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_LBN_DBLCLK(IDC_DOCLIST, OnDbLclkDoclist)
    ON_BN_CLICKED(IDC_PRINT, OnPrint)
    ON_BN_CLICKED(IDC_CLOSE, OnCloseDlg)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

```

```

////////////////////////////////////
// CMainDlg message handlers

BOOL CMainDlg::OnInitDialog()
{
    CListBox    * pList = (CListBox*)GetDlgItem( IDC_DOCLIST );
    CComboBox   * pPrinterList = (CComboBox*)GetDlgItem( IDC_PRINTERS );
    long         l, num_hits;
    char        * pToken;
    PROCESS_INFORMATION    pi;
    STARTUPINFO    sui;
    char           cmd[300], Misc[100];
    BOOL           rc;
    DWORD          id;

    CDialog::OnInitDialog();

    pMainDlg    = this;
    DdeInstance = 0;

    m_DocOpened = FALSE;
    m_DocID     = 0;

    ( (CButton*)GetDlgItem( IDC_PRINT ) )->EnableWindow( FALSE );

    SetIcon(m_hIcon, FALSE);

    // Start up the OnDemand client

    // /I - Enable DDE Interface
    // /W - Window placement (N = hidden)
    // /V - Disable anticipation
    // /B - Disable User Confirmation
    strcpy( cmd, "g:\\ars32\\arsgui32.exe /I /W N /V /B /1 g:\\ars32\\locale\\enu" );

```



```

memset( &sui, 0, sizeof(STARTUPINFO) );
sui.cb = sizeof(STARTUPINFO);
rc = CreateProcess( NULL, cmd, NULL, NULL, FALSE, CREATE_NEW_CONSOLE, NULL, NULL, &sui, &pi );
if ( !rc )
{
    id = GetLastError( );
    FormatMessage( FORMAT_MESSAGE_FROM_SYSTEM | FORMAT_MESSAGE_IGNORE_INSERTS,
                  NULL, id, 0, cmd, sizeof(cmd), NULL );
    sprintf( Misc, "CreateProcessFailed - %s", cmd );
    AfxMessageBox( Misc );
    Misc[0] = '\\0';
}
else
{
    // Start a dde conversation with the client.

    if ( DdeInstance == 0 )
    {
        FARPROC pfnDdeCallBack;
        pfnDdeCallBack = MakeProcInstance( (FARPROC)DdeCallBack, pApp->m_hInstance );
        DdeInitialize( &DdeInstance,
                      (PFNCALLBACK)pfnDdeCallBack,
                      APPCLASS_STANDARD | APPCMD_CLIENTONLY,
                      0L );
    }
    hsz1 = DdeCreateStringHandle( DdeInstance, ARS_DDE_SERVICE, 0 );
    hsz2 = DdeCreateStringHandle( DdeInstance, ARS_DDE_TOPIC, 0 );
    for ( int j = 0; j < 1000; j++ )
    {
        hDdeConv = DdeConnect( DdeInstance, hsz1, hsz2, NULL );
        if ( hDdeConv != NULL )
            break;
    }
    DdeFreeStringHandle( DdeInstance, hsz1 );
    DdeFreeStringHandle( DdeInstance, hsz2 );
    if ( hDdeConv == NULL )
        AfxMessageBox( "Unable to connect to ARSGUI32." );
}

```

```

else
{
    int k;

    // Begin sending dde commands to the client.

    Misc[0] = '/';
    Misc[1] = ARS_DDE_SWITCH_HANDLE;
    sprintf( &Misc[2], "%ld", (long)(char far *)pApp->m_pMainWnd->m_hWnd );
    strcat( Misc, " " );
    strcat( Misc, "/" );
    k = strlen( Misc );
    Misc[k++] = ARS_DDE_SWITCH_CLIENT_NAME;
    strcpy( &Misc[k], "DDE Partner 1" );
    DoDdeCommand( ARS_DDE_CMD_ENABLE_SWITCH_FOCUS, Misc );

    DoDdeCommand( ARS_DDE_CMD_LOGON, "/S gunnar /U demo /P" );

    DoDdeCommand( ARS_DDE_CMD_OPEN_FOLDER, "/F Credit Card Statementss" );

    DoDdeCommand( ARS_DDE_CMD_SEARCH_FOLDER, "" );

    if ( DoDdeCommand( ARS_DDE_CMD_GET_NUM_DOCS_IN_LIST, "" ) )
    {
        num_hits = atoi( RequestedData );
        for ( l = 0; l < num_hits; l++ )
        {
            Misc[0] = '/';
            Misc[1] = ARS_DDE_DOC_NUMBER;
            sprintf( &Misc[2], "%ld", l );

```

```

        if ( DoDdeCommand( ARS_DDE_CMD_GET_DOC_VALUES, Misc ) )
        {
            for ( pToken = strtok( RequestedData, ARS_DDE_DATA_SEPARATOR ),
                  Misc[0] = '\0';
                  pToken != NULL;
                  pToken = strtok( NULL, ARS_DDE_DATA_SEPARATOR ) )
            {
                strcat( Misc, pToken );
                strcat( Misc, " - " );
            }
            if ( Misc[0] != '\0' )
            {
                j = pList->InsertString( -1, Misc );
                pList->SetItemData( j, (DWORD)1 );
            }
        }
        else
            break;
    }
}

DoAdviseLoop( ARS_DDE_ADVISE_LOOP_1, FALSE );
}

if ( DoDdeCommand( ARS_DDE_CMD_GET_PRINTERS, "/L" ) )
{
    for ( pToken = strtok( RequestedData, ARS_DDE_DATA_SEPARATOR );
          pToken != NULL;
          pToken = strtok( NULL, ARS_DDE_DATA_SEPARATOR ) )
        pPrinterList->InsertString( -1, pToken );

    pPrinterList->SetCurSel( 0 );
}

return TRUE;
}

```

```

void CMainDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for painting

        SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

        // Center icon in client rectangle
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
}

// The system calls this to obtain the cursor to display while the user drags
// the minimized window.
HCURSOR CMainDlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}

```

```

void CMainDlg::OnDblickDoclist()
{
    CListBox * pDocsList;
    char        printer[100];
    char        Misc[100];

    pDocsList = (CListBox*)GetDlgItem( IDC_DOCLIST );

    if( m_DocOpened )
    {
        sprintf( Misc, "/D %d", m_DocID );
        DoDdeCommand( ARS_DDE_CMD_CLOSE_DOC, Misc );
    }

    Misc[0] = '/';
    Misc[1] = ARS_DDE_DOC_NUMBER;
    sprintf( &Misc[2], "%d", (int)pDocsList->GetCurSel() );
    if ( DoDdeCommand( ARS_DDE_CMD_OPEN_DOC, Misc ) )
    {
        m_DocID = atol( RequestedData );
        m_DocOpened = TRUE;

        DoDdeCommand( ARS_DDE_CMD_SHOW_WINDOW, "/W" );
    }
    else
    {
        m_DocID = 0;
        m_DocOpened = TRUE;
    }

    GetDlgItem( IDC_PRINTERS )->GetWindowText( printer, sizeof(printer) );
    if( printer != NULL && printer[0] != '\0' )
        ( (CButton*)GetDlgItem( IDC_PRINT ) )->EnableWindow( TRUE );
}

```

```

void CMainDlg::OnPrint()
{
    char printer[100];
    char Misc[100];

    GetDlgItem( IDC_PRINTERS )->GetWindowText( printer, sizeof(printer) );

    Misc[0] = '\\0';
    sprintf( Misc, "/L %s", printer );
    DoDdeCommand( ARS_DDE_CMD_PRINT_DOC, Misc );
}

void CMainDlg::OnCloseDlg()
{
    char Misc[100];

    if( m_DocOpened )
    {
        sprintf( Misc, "/D %d", m_DocID );
        DoDdeCommand( ARS_DDE_CMD_CLOSE_DOC, Misc );
    }

    DoDdeCommand( ARS_DDE_CMD_CLOSE_FOLDER, "" );

    DoDdeCommand( ARS_DDE_CMD_LOGOFF, "" );

    DoDdeCommand( ARS_DDE_CMD_EXIT, "" );

    EndDialog(0);
}

```

Appendix C. Microsoft Visual Basic 5.0 OLE Program Sample

This program sample is provided on an as-is basis. A licensee of the OnDemand product is free to copy, revise, modify, and make derivative works of this program sample as they see fit.

This program was written and compiled using Microsoft Visual Basic 5.0 and demonstrates the use of the following OnDemand OLE control methods:

- CloseDoc
- CloseFolder
- GetDocDisplayValue
- GetNumDocsInList
- Logoff
- Logon
- OpenDoc
- OpenFolder
- ScrollDocHorz
- ScrollDocVert
- SearchFolder
- SetDocZoom
- SetFolderSearchFieldData
- SetUserMessageMode

Global Variables Used by the Demo

Option Explicit

Global Const defini = "vbarsole.ini" 'Default ini file name

Global Const defstanza = "VBARSOLE" 'Default stanza

' The following constants were obtained from arsoleex.h

Global Const ARS_OLE_USER_MSG_MODE_SHOW = 1

Global Const ARS_OLE_USER_MSG_MODE_SUPPRESS = 2

Global Const ARS_OLE_FIND_FIRST = 1

Global Const ARS_OLE_FIND_PREV = 2

Global Const ARS_OLE_FIND_NEXT = 3

Global Const ARS_OLE_OPR_EQUAL = 1

Global Const ARS_OLE_OPR_NOT_EQUAL = 2

Global Const ARS_OLE_OPR_LESS_THAN = 3

Global Const ARS_OLE_OPR_LESS_THAN_OR_EQUAL = 4

Global Const ARS_OLE_OPR_GREATER_THAN = 5

Global Const ARS_OLE_OPR_GREATER_THAN_OR_EQUAL = 6

Global Const ARS_OLE_OPR_BETWEEN = 7

Global Const ARS_OLE_OPR_NOT_BETWEEN = 8

Global Const ARS_OLE_OPR_IN = 9

Global Const ARS_OLE_OPR_NOT_IN = 10

Global Const ARS_OLE_OPR_LIKE = 11

Global Const ARS_OLE_OPR_NOT_LIKE = 12

```

Global Const ARS_OLE_RC_SUCCESS = 0
Global Const ARS_OLE_RC_NO_MEMORY = 1
Global Const ARS_OLE_RC_SERVER_ERROR = 2
Global Const ARS_OLE_RC_USER_CANCELLED = 3
Global Const ARS_OLE_RC_INVALID_DIRECTORY = 4
Global Const ARS_OLE_RC_UNAUTHORIZED_OPERATION = 5
Global Const ARS_OLE_RC_NOT_SUPPORTED = 6
Global Const ARS_OLE_RC_FILE_ERROR = 7
Global Const ARS_OLE_RC_ALREADY_LOGGED_ON = 8
Global Const ARS_OLE_RC_NOT_LOGGED_ON = 9
Global Const ARS_OLE_RC_FOLDER_ALREADY_OPEN = 10
Global Const ARS_OLE_RC_FOLDER_NOT_OPEN = 11
Global Const ARS_OLE_RC_UNKNOWN_FOLDER = 12
Global Const ARS_OLE_RC_NO_FOLDERS_AVAILABLE = 13
Global Const ARS_OLE_RC_DOC_NOT_OPEN = 14
Global Const ARS_OLE_RC_DOC_ALREADY_OPEN = 15
Global Const ARS_OLE_RC_NO_DOC_AVAILABLE = 16
Global Const ARS_OLE_RC_OPEN_DOC_FAILED = 17
Global Const ARS_OLE_RC_DOC_CANNOT_HORZ_SCROLL = 18
Global Const ARS_OLE_RC_INVALID_DOC_INDEX = 19
Global Const ARS_OLE_RC_INVALID_CONTROL_ID = 20
Global Const ARS_OLE_RC_INVALID_FIELD = 21
Global Const ARS_OLE_RC_INVALID_OPERATOR = 22
Global Const ARS_OLE_RC_INVALID_MESSAGE_MODE = 23
Global Const ARS_OLE_RC_INVALID_ZOOM_PERCENT = 24
Global Const ARS_OLE_RC_INVALID_PAGE_NUMBER = 25
Global Const ARS_OLE_RC_INVALID_ROTATION = 26
Global Const ARS_OLE_RC_INVALID_COLOR = 27
Global Const ARS_OLE_RC_INVALID_COPIES = 28
Global Const ARS_OLE_RC_INVALID_ORIENTATION = 29
Global Const ARS_OLE_RC_INVALID_PRINTER = 30
Global Const ARS_OLE_RC_INVALID_FIND_TYPE = 31
Global Const ARS_OLE_RC_ERROR_DURING_PRINT = 32

Global Const ARS_OLE_SCROLL_LINEUP = 0
Global Const ARS_OLE_SCROLL_LINELEFT = 0
Global Const ARS_OLE_SCROLL_LINEDOWN = 1
Global Const ARS_OLE_SCROLL_LINERIGHT = 1
Global Const ARS_OLE_SCROLL_PAGEUP = 2
Global Const ARS_OLE_SCROLL_PAGELEFT = 2
Global Const ARS_OLE_SCROLL_PAGEDOWN = 3
Global Const ARS_OLE_SCROLL_PAGERIGHT = 3
Global Const ARS_OLE_SCROLL_THUMBPOSITION = 4
Global Const ARS_OLE_SCROLL_THUMBTRACK = 5
Global Const ARS_OLE_SCROLL_TOP = 6
Global Const ARS_OLE_SCROLL_LEFT = 6
Global Const ARS_OLE_SCROLL_BOTTOM = 7
Global Const ARS_OLE_SCROLL_RIGHT = 7
Global Const ARS_OLE_SCROLL_ENDSCROLL = 8

```

```
Global Const DocZoom = 110

Global server As String      'Server name
Global userid As String      'userid
Global password As String    'password
Global folder As String      'folder

Global doc_id As Integer
Global doc_values(0 To 8) As String

Global OpenDoc As Boolean
Global VertScroll101d As Integer
Global HorzScroll101d As Integer
```

```

'Define the Windows APIs used by the program
Declare Function GetPrivateProfileInt Lib "kernel32" Alias "GetPrivateProfileIntA"
    (ByVal lpApplicationName As String, ByVal lpKeyName As String, ByVal nDefault As Long,
    ByVal lpFileName As String) As Long
Declare Function GetPrivateProfileString Lib "kernel32" Alias "GetPrivateProfileStringA"
    (ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpDefault As String,
    ByVal lpReturnedString As String, ByVal nSize As Long, ByVal lpFileName As String) As Long
Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
'Declare Function GetPrivateProfileString Lib "kernel32" (ByVal sname$, ByVal Kname$, ByVal Def$,
    ByVal ret$, ByVal Size%, ByVal FName$) As Integer

Public Sub Main()
    Dim rc As Integer

    Load frmMain
    Load frmInit

    doc_id = 0
    OpenDoc = False
    VertScroll10ld = 0
    HorzScroll10ld = 0

    'Disable "View" buttons
    frmMain.cmdViewStmt1.Enabled = False
    frmMain.cmdViewStmt2.Enabled = False
    frmMain.cmdViewStmt3.Enabled = False

    'Because we need the ocx file and the arssck32.dll
    'which reside in the ars directory I will require
    'that this exe and its ini file also reside in the
    'ars install directory.

    'I should check for ini file existance first.

    'Try to find the "Server" name in the ini file
    server = fncParmGet(defstanza, "Server", defini)
    If Len(server) = 0 Then
        MsgBox "Cannot find Server in " + defini
    End
End If

```

```

'Try to find the "Userid" name in the ini file
userid = fncParmGet(defstanza, "Userid", defini)
If Len(userid) = 0 Then
    MsgBox "Cannot find Userid in " + defini
End
End If

'Try to find the "Server" name in the ini file
password = fncParmGet(defstanza, "Password", defini)
If Len(password) = 0 Then
    password = " "
End If

'Try to find the "Folder" name in the ini file
folder = fncParmGet(defstanza, "Folder", defini)
If Len(folder) = 0 Then
    MsgBox "Cannot find Folder in " + defini
End
End If

'The following call is for debug.
rc = frmMain.ArsOle.SetUserMessageMode(ARS_OLE_USER_MSG_MODE_SHOW)

frmInit.Show
frmInit.pnlStatus.Caption = "Logging on to Server..."

'Attempt to logon to the specified server.
rc = frmMain.ArsOle.Logon(server, userid, password)
If rc <> ARS_OLE_RC_SUCCESS Then
    frmInit.pnlStatus.Caption = ""
    MsgBox "Cannot Logon to server " + server + "; rc = " + Str(rc)
End
End If

frmInit.SetFocus

'Attempt to open the folder specified in the ini file.
frmInit.pnlStatus.Caption = "Opening folder..."
rc = frmMain.ArsOle.OpenFolder(folder)
If rc <> ARS_OLE_RC_SUCCESS Then
    frmMain.pnlStatus.Caption = ""
    MsgBox "Cannot open folder " + folder + "; rc = " + Str(rc)
    frmMain.ArsOle.Logoff
End
End If

frmInit.SetFocus
frmInit.pnlStatus.Caption = ""
frmInit.Hide

frmMain.Show
End Sub

```

```

'This function returns information from the ini file.
Function fncParmGet(ByVal stanza As String, ByVal keyname As String, ByVal inifile As String)
    Dim Default, result As String
    Dim rc As Integer

    Default = ""
    result = Space$(255)

    rc = GetPrivateProfileString(stanza, keyname, Default, result, Len(result), inifile)
    If rc Then
        fncParmGet = Trim$(result)
        If Len(fncParmGet) > 1 Then
            fncParmGet = Left$(fncParmGet, Len(fncParmGet) - 1)
        End If
    Else
        fncParmGet = ""
    End If

End Function

'This function is only used to dummy up the date paid
'field of the form because for the
'demo, which uses the 'Baxter Bay Credit' folder,
'we cannot get this information from the database.
'This function adds approximately 20 days to the statement
'date field (which is passed in).
Public Function fncParseDate(ByVal stmtdate As String)
    Dim date_array(1 To 3) As String
    Dim searchch, workline, workchar As String
    Dim txtptr, lenstring, i As Integer
    Dim pay_day, pay_month, pay_year As Integer

    txtptr = 0
    searchch = Chr(47)
    workline = ""
    lenstring = Len(stmtdate)

```

```

'Extract chars to the first '/'
For i = 1 To lenstring
    workchar = Mid$(stmtdate, i, 1)
    'When a '/' is found, store result, reset
    If workchar = searchch Then
        txtptr = txtptr + 1
        date_array(txtptr) = workline
        workline = ""
    'Otherwise, keep building the work string
    Else
        workline = workline + workchar
    End If
Next

If Len(workline) > 0 Then
    txtptr = txtptr + 1
    date_array(txtptr) = workline
End If

'date_array contains three elements, the first is the month
'number, the second is the day of the month, and the third is
'the year. Simply check if the day of the month plus 20
'is greater than 28, if so the difference becomes the new
'day of the month, and we increment the month number.
pay_day = Int(date_array(2)) + 20
pay_month = Int(date_array(1))
pay_year = Int(date_array(3))
If pay_day > 28 Then
    pay_day = pay_day - 28
    pay_month = pay_month + 1
    If pay_month > 12 Then
        pay_month = 1
        pay_year = pay_year + 1
    End If
End If

fncParseDate = LTrim(Str(pay_month)) + "/" + LTrim(Str(pay_day)) + "/" + LTrim(Str(pay_year))
End Function

```



```

Private Sub cmdCustInfo_Click()
    Dim rc As Integer
    Dim acct_num, ini_str As String
    Dim first_num, second_num, third_num As Integer
    Dim temp As String
    Dim numdocs As Variant

    If OpenDoc Then
        pnlStatus.Caption = "Closing document..."
        rc = ArsOle.CloseDoc()
        pnlStatus.Caption = ""
    End If

    'Clear the payment record fields
    pnlStmtDate1.Caption = ""
    pnlStmtDate2.Caption = ""
    pnlStmtDate3.Caption = ""
    pnlBalance1.Caption = ""
    pnlBalance2.Caption = ""
    pnlBalance3.Caption = ""
    pnlDatePaid1.Caption = ""
    pnlDatePaid2.Caption = ""
    pnlDatePaid3.Caption = ""

    'Clear the customer information fields
    pnlNameData = ""
    pnlSOSData = ""
    pnlDOBData = ""
    pnlMNameData = ""
    pnlAddrData1 = ""
    pnlAddrData2 = ""
    pnlPhoneData = ""

```

```

'Disable "View" buttons
cmdViewStmt1.Enabled = False
cmdViewStmt2.Enabled = False
cmdViewStmt3.Enabled = False

'Look up the account number, contained in the pnlAcctnumData text field
'in the arsvblan.ini file. If found, read the respective
'fields. If not found display error message.
acct_num = txtAcctnumData.Text

'Do at least a little validation.
If Len(acct_num) <> 11 Then
    MsgBox "Correct format for account # is 000-000-000"
    Exit Sub
End If

'If we have gotten to here we know that we have an account
'number of the format 000-000-000. If either of the first
'two sections of the number are non-zero or if the third
'section is not between 001-046 then default to the account
'number 000-000-001.
first_num = Int(Mid(acct_num, 1, 3))
second_num = Int(Mid(acct_num, 5, 3))
third_num = Int(Mid(acct_num, 9, 3))
If first_num <> 0 Or second_num <> 0 Or third_num > 46 Then
    acct_num = "000-000-001"
ElseIf third_num = 0 Then
    MsgBox "Invalid account number!"
    Exit Sub
End If

```

```

ini_str = fncParmGet(acct_num, "Name", defini)
If Len(ini_str) = 0 Then
    MsgBox "'Name' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlNameData.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "SSN", defini)
If Len(ini_str) = 0 Then
    MsgBox "'SSN' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlSSNData.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "DOB", defini)
If Len(ini_str) = 0 Then
    MsgBox "'DOB' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlDOBData.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "MaidenName", defini)
If Len(ini_str) = 0 Then
    MsgBox "'MaidenName' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlMNameData.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "StreetAddress", defini)
If Len(ini_str) = 0 Then
    MsgBox "'StreetAddress' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlAddrData1.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "CityStateZip", defini)
If Len(ini_str) = 0 Then
    MsgBox "'CityStateZip' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlAddrData2.Caption = " " + ini_str

ini_str = fncParmGet(acct_num, "PhoneNum", defini)
If Len(ini_str) = 0 Then
    MsgBox "'PhoneNum' field not found for acct#" + acct_num + "in " + ininame
    Exit Sub
End If
pnlPhoneData.Caption = " " + ini_str

```

```

'We are changing customer accounts so before we get new customer
'information, close old customers open documents.
If doc_id <> 0 Then
    rc = ArsOle.CloseDoc
    If rc <> ARS_OLE_RC_SUCCESS Then
        pnlStatus.Caption = ""
        MsgBox "Cannot set folder search criteria; rc = " + Str(rc)
        ArsOle.CloseFolder
        ArsOle.Logoff
    End
End If

pnlStatus.Caption = "Searching folder..."
rc = ArsOle.SetFolderSearchFieldData("Account", ARS_OLE_OPR_EQUAL, acct_num, "")

If rc <> ARS_OLE_RC_SUCCESS Then
    pnlStatus.Caption = ""
    MsgBox "Cannot set folder search criteria; rc = " + Str(rc)
    ArsOle.CloseFolder
    ArsOle.Logoff
End
End If

rc = ArsOle.SearchFolder(0)
If rc <> ARS_OLE_RC_SUCCESS Then
    pnlStatus.Caption = ""
    MsgBox "Search folder failed; rc = " + Str(rc)
    ArsOle.CloseFolder
    ArsOle.Logoff
End
End If

rc = ArsOle.GetNumDocsInList(numdocs)

rc = ArsOle.GetDocDisplayValue(numdocs - 1, 0, temp)
pnlStmtDate1.Caption = temp
pnlDatePaid1.Caption = fncParseDate(temp)
rc = ArsOle.GetDocDisplayValue(numdocs - 2, 0, temp)
pnlStmtDate2.Caption = temp
pnlDatePaid2.Caption = fncParseDate(temp)
rc = ArsOle.GetDocDisplayValue(numdocs - 3, 0, temp)
pnlStmtDate3.Caption = temp
pnlDatePaid3.Caption = fncParseDate(temp)

rc = ArsOle.GetDocDisplayValue(numdocs - 1, 3, temp)
pnlBalance1.Caption = temp
rc = ArsOle.GetDocDisplayValue(numdocs - 2, 3, temp)
pnlBalance2.Caption = temp
rc = ArsOle.GetDocDisplayValue(numdocs - 3, 3, temp)
pnlBalance3.Caption = temp

```

```

        'Enable "View" buttons
        cmdViewStmt1.Enabled = True
        cmdViewStmt2.Enabled = True
        cmdViewStmt3.Enabled = True

        pnlStatus.Caption = ""
End Sub

Private Sub cmdExit_Click()
    'If OpenDoc Then
    '    ArsOle.CloseDoc
    'End If
    'ArsOle.CloseFolder
    'ArsOle.Logoff
End Sub

Private Sub cmdViewStmt1_Click()
    Dim numdocs As Variant

    rc = ArsOle.GetNumDocsInList(numdocs)

    If OpenDoc Then
        pnlStatus.Caption = "Closing document..."
        rc = ArsOle.CloseDoc()
        pnlStatus.Caption = ""
        vscrollDoc.Value = 0
        hscrollDoc.Value = 0
    End If

    pnlStatus.Caption = "Retrieving document..."
    rc = ArsOle.OpenDoc(numdocs - 1, "", 0)
    If rc <> ARS_OLE_RC_SUCCESS Then
        pnlStatus.Caption = ""
        MsgBox "Open document failed; rc = " + Str(rc)
        ArsOle.CloseFolder
        ArsOle.Logoff
    End If
    End If
    pnlStatus.Caption = ""

    OpenDoc = True

    rc = ArsOle.SetDocZoom(DocZoom, horzPos, vertPos)
    vscrollDoc.Value = vertPos
    hscrollDoc.Value = horzPos
End Sub

```

```

Private Sub cmdViewStmt2_Click()
    Dim numdocs As Variant

    rc = ArsOle.GetNumDocsInList(numdocs)

    If OpenDoc Then
        pnlStatus.Caption = "Closing document..."
        rc = ArsOle.CloseDoc()
        pnlStatus.Caption = ""
        vscrollDoc.Value = 0
        hscrollDoc.Value = 0
    End If

    pnlStatus.Caption = "Retrieving document..."
    rc = ArsOle.OpenDoc(numdocs - 2, "", 0)
    If rc <> ARS_OLE_RC_SUCCESS Then
        pnlStatus.Caption = ""
        MsgBox "Open document failed; rc = " + Str(rc)
        ArsOle.CloseFolder
        ArsOle.Logoff
        End
    End If
    pnlStatus.Caption = ""

    OpenDoc = True

    rc = ArsOle.SetDocZoom(DocZoom, horzPos, vertPos)
End Sub

```

```

Private Sub cmdViewStmt3_Click()
    Dim numdocs As Variant

    rc = ArsOle.GetNumDocsInList(numdocs)

    If OpenDoc Then
        pnlStatus.Caption = "Closing document..."
        rc = ArsOle.CloseDoc()
        pnlStatus.Caption = ""
        vscrollDoc.Value = 0
        hscrollDoc.Value = 0
    End If

    pnlStatus.Caption = "Retrieving document..."
    rc = ArsOle.OpenDoc(numdocs - 3, "", 0)
    If rc <> ARS_OLE_RC_SUCCESS Then
        pnlStatus.Caption = ""
        MsgBox "Open document failed; rc = " + Str(rc)
        ArsOle.CloseFolder
        ArsOle.Logoff
    End
End If
pnlStatus.Caption = ""

OpenDoc = True

rc = ArsOle.SetDocZoom(DocZoom, horzPos, vertPos)
End Sub

Private Sub Form_Unload(Cancel As Integer)
    If OpenDoc Then
        ArsOle.CloseDoc
    End If
    ArsOle.CloseFolder
    ArsOle.Logoff
End
End Sub

```

```

Private Sub hscrollDoc_Change()
    Dim Diff As Integer
    Dim rc As Integer
    Dim ScrollCode As Integer
    Dim NewPos As Variant

    NewPos = 0
    Diff = hscrollDoc.Value - HorzScrollOld
    If Diff = hscrollDoc.LargeChange Then
        ScrollCode = ARS_OLE_SCROLL_PAGERIGHT
        rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
        hscrollDoc.Value = NewPos
    ElseIf Diff = -hscrollDoc.LargeChange Then
        ScrollCode = ARS_OLE_SCROLL_PAGELEFT
        rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
        hscrollDoc.Value = NewPos
    ElseIf Diff = hscrollDoc.SmallChange Then
        ScrollCode = ARS_OLE_SCROLL_LINERIGHT
        rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
        hscrollDoc.Value = NewPos
    ElseIf Diff = -hscrollDoc.SmallChange Then
        ScrollCode = ARS_OLE_SCROLL_LINELEFT
        rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
        hscrollDoc.Value = NewPos
    Else
        ScrollCode = ARS_OLE_SCROLL_THUMBPOSITION
        NewPos = hscrollDoc.Value
        rc = ArsOle.ScrollDocHorz(ScrollCode, NewPos)
        HorzScrollOld = hscrollDoc.Value
    End If

    HorzScrollOld = hscrollDoc.Value
End Sub

```



```

Private Sub vscrollDoc_Change()
    Dim Diff As Integer
    Dim rc As Integer
    Dim ScrollCode As Integer
    Dim NewPos As Variant

    NewPos = 0
    Diff = vscrollDoc.Value - VertScroll101d
    If Diff = vscrollDoc.LargeChange Then
        ScrollCode = ARS_OLE_SCROLL_PAGEDOWN
        rc = Ars01e.ScrollDocVert(ScrollCode, NewPos)
        VertScroll101d = NewPos
        vscrollDoc.Value = NewPos
    ElseIf Diff = -vscrollDoc.LargeChange Then
        ScrollCode = ARS_OLE_SCROLL_PAGEUP
        rc = Ars01e.ScrollDocVert(ScrollCode, NewPos)
        VertScroll101d = NewPos
        vscrollDoc.Value = NewPos
    ElseIf Diff = vscrollDoc.SmallChange Then
        ScrollCode = ARS_OLE_SCROLL_LINEDOWN
        rc = Ars01e.ScrollDocVert(ScrollCode, NewPos)
        VertScroll101d = NewPos
        vscrollDoc.Value = NewPos
    ElseIf Diff = -vscrollDoc.SmallChange Then
        ScrollCode = ARS_OLE_SCROLL_LINEUP
        rc = Ars01e.ScrollDocVert(ScrollCode, NewPos)
        VertScroll101d = NewPos
        vscrollDoc.Value = NewPos
    Else
        ScrollCode = ARS_OLE_SCROLL_THUMBPOSITION
        NewPos = vscrollDoc.Value
        rc = Ars01e.ScrollDocVert(ScrollCode, NewPos)
        VertScroll101d = vscrollDoc.Value
    End If
End Sub

```

Appendix D. Microsoft VC++ 5.0 OLE Program Sample

This program sample is provided on an as-is basis. A licensee of the OnDemand product is free to copy, revise, modify, and make derivative works of this program sample as they see fit.

This program was written and compiled using Microsoft VC++ 5.0 and demonstrates the use of the following OnDemand OLE control methods:

```
CloseDoc
CloseFolder
GetDocBackgroundColor
GetDocDisplayValues
GetDocImageColor
GetDocNumPages
GetDocRotation
GetDocZoom
GetNumDocsInList
GetNumFolderDisplayFields
IsDocHorzScrollRequired
Logoff
Logon
OpenDoc
OpenFolder
PrintDoc
SearchFolder
ScrollDocHorz
ScrollDocVert
SetDocBackgroundColor
SetDocImageColor
SetDocRotation
SetDocZoom
SetUserMessageMode

#include "stdafx.h"
#include <winpool.h>

#include "vcole32.h"
#include "MainDlg.h"
#include "AttrsDlg.h"

static CMainDlg * pMainDlg;

#define COLOR_MAP struct _ColorMap
COLOR_MAP
{
    short    color;
    char * pText;
};
```

```

static COLOR_MAP Colors[] =
{ { ARS_OLE_COLOR_BLACK, "Black" },
  { ARS_OLE_COLOR_WHITE, "White" },
  { ARS_OLE_COLOR_RED, "Red" },
  { ARS_OLE_COLOR_BLUE, "Blue" },
  { ARS_OLE_COLOR_GREEN, "Green" },
  { ARS_OLE_COLOR_YELLOW, "Yellow" },
  { ARS_OLE_COLOR_GREY, "Grey" },
  { ARS_OLE_COLOR_CYAN, "Cyan" },
  { ARS_OLE_COLOR_MAGENTA, "Magenta" } };

#define NUM_COLORS ( sizeof(Colors) / sizeof(COLOR_MAP) )

#define ERROR_MAP struct _ErrorMap
ERROR_MAP
{
    short code;
    char * pMsg;
};

static ERROR_MAP Errors[] =
{ { ARS_OLE_RC_NO_MEMORY, "insufficient memory" },
  { ARS_OLE_RC_UNKNOWN_FOLDER, "unknown folder" },
  { ARS_OLE_RC_NO_FOLDERS_AVAILABLE, "no folders availble" },
  { ARS_OLE_RC_SERVER_ERROR, "server error" },
  { ARS_OLE_RC_FOLDER_ALREADY_OPEN, "folder already open" },
  { ARS_OLE_RC_NOT_LOGGED_ON, "not logged on" },
  { ARS_OLE_RC_ALREADY_LOGGED_ON, "already logged on" },
  { ARS_OLE_RC_INVALID_DIRECTORY, "invalid directory" },
  { ARS_OLE_RC_FOLDER_NOT_OPEN, "folder not open" },
  { ARS_OLE_RC_DOC_ALREADY_OPEN, "document already open" },
  { ARS_OLE_RC_DOC_NOT_OPEN, "no document is open" },
  { ARS_OLE_RC_OPEN_DOC_FAILED, "open doc failed" },
  { ARS_OLE_RC_UNAUTHORIZED_OPERATION, "unauthorized operation" },
  { ARS_OLE_RC_USER_CANCELLED, "user cancelled operation" },
  { ARS_OLE_RC_INVALID_INDEX, "invalid index" },
  { ARS_OLE_RC_INVALID_FIELD, "invalid field" },
  { ARS_OLE_RC_INVALID_OPERATOR, "invalid operator" },
  { ARS_OLE_RC_INVALID_MESSAGE_MODE, "invalid message mode" },
  { ARS_OLE_RC_INVALID_ZOOM_PERCENT, "invalid zoom percent" },
  { ARS_OLE_RC_DOC_CANNOT_HORZ_SCROLL, "cannot horz scroll" },
  { ARS_OLE_RC_INVALID_PAGE_NUMBER, "invalid page number" },
  { ARS_OLE_RC_INVALID_CONTROL_ID, "invalid other control" },
  { ARS_OLE_RC_INVALID_ROTATION, "invalid rotation" },
  { ARS_OLE_RC_NO_DOC_AVAILABLE, "no document for hit" },
  { ARS_OLE_RC_NOT_SUPPORTED, "not supported" },
  { ARS_OLE_RC_FILE_ERROR, "file error" },
  { ARS_OLE_RC_INVALID_COPIES, "invalid copies" },
  { ARS_OLE_RC_INVALID_ORIENTATION, "invalid orientation" },
  { ARS_OLE_RC_INVALID_PRINTER, "invalid printer" },
  { ARS_OLE_RC_INVALID_FIND_TYPE, "invalid find type" },
  { ARS_OLE_RC_ERROR_DURING_PRINT, "error during print" },
  { ARS_OLE_RC_INVALID_COLOR, "invalid color" } };

```

```

#define NUM_ERRORS ( sizeof(Errors) / sizeof(ERROR_MAP) )

BEGIN_MESSAGE_MAP(CMainDlg, CDialog)
    //{AFX_MSG_MAP(CMainDlg)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDC_PRINT, OnPrint)
    ON_LBN_DBLCLK(IDC_DOCLIST, OnDbldclkDoclist)
    ON_BN_CLICKED(IDC_CLOSE, OnCloseDlg)
    ON_WM_HSCROLL()
    ON_WM_VSCROLL()
    ON_WM_CLOSE()
    ON_BN_CLICKED(IDC_ATTRIBUTES, OnSetDocAttrs)
    //}AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_EVENTSINK_MAP(CMainDlg, CDialog)
    //{AFX_EVENTSINK_MAP(CMainDlg)
    ON_EVENT(CMainDlg, IDC_ARSCtrl, 4, OnFolderSearchCompletedArscrtl, VTS_NONE)
    ON_EVENT(CMainDlg, IDC_ARSCtrl, 3, OnDocOpenedArscrtl, VTS_NONE)
    ON_EVENT(CMainDlg, IDC_ARSCtrl, 1, OnDocClosedArscrtl, VTS_NONE)
    //}AFX_EVENTSINK_MAP
END_EVENTSINK_MAP()

CMainDlg::CMainDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CMainDlg::IDD, pParent)
{
    //{AFX_DATA_INIT(CMainDlg)
    // NOTE: the ClassWizard will add member initialization here
    //}AFX_DATA_INIT
    // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

CMainDlg:: CMainDlg()
{
}

void CMainDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{AFX_DATA_MAP(CMainDlg)
    // NOTE: the ClassWizard will add DDX and DDV calls here
    //}AFX_DATA_MAP
}

```

```

////////////////////////////////////
// CMainDlg message handlers

BOOL CMainDlg::OnInitDialog()
{
    VARIANT    var;
    short      rc;
    char       Misc[1024];
    CArsOle * pArsCtrl;
    int        index;

    pMainDlg = this;

    m_DocOpened = FALSE;

    ( (CButton*)GetDlgItem( IDC_PRINT ) )->EnableWindow( FALSE );
    ( (CButton*)GetDlgItem( IDC_ATTRIBUTES ) )->EnableWindow( FALSE );

    SetIcon(m_hIcon, FALSE);

    pArsCtrl = (CArsOle*)GetDlgItem( IDC_ARSCtrl );

    ( (CScrollbar*)GetDlgItem( IDC_HORZ_SCROLLBAR ) )->SetScrollRange( 0, ARS_OLE_SCROLL_RANGE );
    ( (CScrollbar*)GetDlgItem( IDC_VERT_SCROLLBAR ) )->SetScrollRange( 0, ARS_OLE_SCROLL_RANGE );
    ( (CScrollbar*)GetDlgItem( IDC_HORZ_SCROLLBAR ) )->ShowScrollBar( FALSE );
    ( (CScrollbar*)GetDlgItem( IDC_VERT_SCROLLBAR ) )->ShowScrollBar( FALSE );

    // Begin calling functions in the OnDemand OLE control

    rc = pArsCtrl->SetUserMessageMode( ARS_OLE_USER_MSG_MODE_SHOW );
    if ( rc != ARS_OLE_RC_SUCCESS )
        DisplayMsg( rc, "SetUserMessageMode" );

    rc = pArsCtrl->Logon( "gunnar", "demo", " " );
    if ( rc != ARS_OLE_RC_SUCCESS )
        DisplayMsg( rc, "Logon" );

    rc = pArsCtrl->OpenFolder( "Credit Card Statements" );
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        DisplayMsg( rc, "OpenFolder" );
        return FALSE;
    }
}

```

```

rc = pArsCtrl->GetNumFolderDisplayFields( &var );
if ( rc != ARS_OLE_RC_SUCCESS )
{
    DisplayMsg( rc, "GetNumFolderDisplayFields" );
    return FALSE;
}
m_NumDisplayFields = var.iVal;

rc = pArsCtrl->SearchFolder( FALSE );
if ( rc != ARS_OLE_RC_SUCCESS )
{
    DisplayMsg( rc, "SearchFolder" );
    return FALSE;
}

// Get the list of local printers
CComboBox * pPrintersList;
PRINTER_INFO_2 * pPrinterInfoArray;
DWORD size, num_printer_infos, printer_info_index, port_index;
char * pPortNames, * pIndividualPortName;
pPrintersList = (CComboBox*)GetDlgItem( IDC_PRINTERS );
EnumPrinters( PRINTER_ENUM_LOCAL | PRINTER_ENUM_CONNECTIONS,
    NULL, 2, NULL, 0, &size, &num_printer_infos );
pPrinterInfoArray = (PRINTER_INFO_2*)new char[ size ];
EnumPrinters( PRINTER_ENUM_LOCAL | PRINTER_ENUM_CONNECTIONS,
    NULL, 2, (BYTE*)pPrinterInfoArray, size, &size, &num_printer_infos );
if ( num_printer_infos > 0 )
{
    for ( printer_info_index = 0;
        printer_info_index < num_printer_infos; printer_info_index++ )
    {
        pPortNames =
            new char[ strlen( pPrinterInfoArray[printer_info_index].pPortName ) + 1 ];
        strcpy( pPortNames, pPrinterInfoArray[printer_info_index].pPortName );
        for ( pIndividualPortName = strtok( pPortNames, "," ), port_index = 0;
            pIndividualPortName != NULL;
            pIndividualPortName = strtok( NULL, "," ), port_index++ )
        {
            strcpy( Misc, pPrinterInfoArray[printer_info_index].pPrinterName );
            strcat( Misc, " on " );
            strcat( Misc, pIndividualPortName );
            index = pPrintersList->AddString( Misc );
        }
        delete pPortNames;
    }
    pPrintersList->SetCurSel( 0 );
}
delete [] pPrinterInfoArray;

return TRUE;
}

```

```

void CMainDlg::DisplayMsg( short rc, char * pMsg )
{
    int    j;
    char   Misc[1024];

    if ( rc == ARS_OLE_RC_SUCCESS )
        AfxMessageBox( pMsg );
    else
    {
        for ( j = 0; j < NUM_ERRORS; j++ )
            if ( Errors[j].code == rc )
                break;
        sprintf( Misc, "%s returned '%s'.", pMsg, j < NUM_ERRORS
                ? Errors[j].pMsg
                : "***INVALID RETURN CODE***" );
        AfxMessageBox( Misc );
    }
}

void CMainDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
    CDialog::OnSysCommand(nID, lParam);
}

void CMainDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for painting

        SendMessage(WM_ICONERASEBKGND, (LPARAM) dc.GetSafeHdc(), 0);

        // Center icon in client rectangle
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
}

```



```

// The system calls this to obtain the cursor to display while the user drags
// the minimized window.
HCURSOR CMainDlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}

void CMainDlg::OnPrint()
{
    CArsOle * pArsCtrl;
    CListBox * pDocsList;
    short      rc;
    char        printer[100];

    pArsCtrl = (CArsOle*)GetDlgItem( IDC_ARSCtrl );
    pDocsList = (CListBox*)GetDlgItem( IDC_DOCLIST );

    GetDlgItem( IDC_PRINTERS )->GetWindowText( printer, sizeof(printer) );

    rc = pArsCtrl->PrintDoc (-1,      // the open doc
                           0,        // entire document
                           printer,
                           1,        // local printer (not server)
                           1,        // # of copies
                           ARS_OLE_ORIENTATION_PORTRAIT,
                           .5, .5, .5, .5, // margins
                           0);        // margins in inches
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        DisplayMsg( rc, "PrintDoc" );
        return;
    }
}

```

```

void CMainDlg::OnDblickDoclist()
{
    CArsOle * pArsCtrl;
    CListBox * pDocsList;
    short      index;
    short      rc;
    char        printer[100];

    pArsCtrl = (CArsOle*)GetDlgItem( IDC_ARSCtrl );
    pDocsList = (CListBox*)GetDlgItem( IDC_DOCLIST );

    if( m_DocOpened )
        pArsCtrl->CloseDoc();

    index = pDocsList->GetCurSel();

    rc = pArsCtrl->OpenDoc( index, NULL, 0 );
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        DisplayMsg( rc, "OpenDoc" );
        return;
    }

    m_DocOpened = TRUE;

    GetDlgItem( IDC_PRINTERS )->GetWindowText( printer, sizeof(printer) );
    if( printer != NULL && printer[0] != '\0' )
        ( CButton*)GetDlgItem( IDC_PRINT )->EnableWindow( TRUE );

    ( CButton*)GetDlgItem( IDC_ATTRIBUTES )->EnableWindow( TRUE );
}

void CMainDlg::OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    VARIANT      var;
    CArsOle * pArsCtrl;
    short      rc;

    pArsCtrl = (CArsOle*)GetDlgItem( IDC_ARSCtrl );

    var.vt = VT_I2;
    var.iVal = nPos;
    rc = pArsCtrl->ScrollDocHorz( (short)nSBCode, &var );
    if ( rc != ARS_OLE_RC_SUCCESS )
        DisplayMsg( rc, "ScrollDocHorz" );
    else
        pScrollBar->SetScrollPos( var.iVal );
}

```

```

void CMainDlg::OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    VARIANT    var;
    CArsOle    * pArsCtrl;
    short      rc;

    pArsCtrl = (CArsOle*)GetDlgItem( IDC_ARSCtrl );

    var.vt = VT_I2;
    var.iVal = nPos;
    rc = pArsCtrl->ScrollDocVert( (short)nSBCode, &var );
    if ( rc != ARS_OLE_RC_SUCCESS )
        DisplayMsg( rc, "ScrollDocVert" );
    else
        pScrollBar->SetScrollPos( var.iVal );
}

void CMainDlg::RefreshDocList( )
{
    VARIANT    var;
    CArsOle    * pArsCtrl;
    ArsOleValue * pValues;
    CListBox    * pDocsList;
    char        temp[21];
    short      rc;
    long        num_docs = 0, j;

    pArsCtrl = (CArsOle*)GetDlgItem( IDC_ARSCtrl );
    pDocsList = (CListBox*)GetDlgItem( IDC_DOCLIST );

    if ( pArsCtrl == NULL || pDocsList == NULL )
        return;
}

```

```

rc = pArsCtrl->GetNumDocsInList( &var );
if ( rc != ARS_OLE_RC_SUCCESS )
{
    DisplayMsg( rc, "GetNumDocsInList" );
    return;
}
num_docs = var.lVal;

pValues = new ArsOleValue[ max( m_NumDisplayFields, 1 ) ];
pDocsList->ResetContent( );

for ( j = 0; j < num_docs; j++ )
{
    rc = pArsCtrl->GetDocDisplayValues( j, (IUnknown*)pValues, m_NumDisplayFields );
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        DisplayMsg( rc, "GetDocDisplayValues" );
        break;
    }
    sprintf( temp, "%s\t%s\t%s", pValues[0], pValues[3], pValues[2] );
    pDocsList->InsertString( -1, temp );
}
pDocsList->SetCurSel( 0 );

delete [] pValues;
}

```

```

void CMainDlg::OnFolderSearchCompletedArsctrl()
{
    RefreshDocList( );
}

void CMainDlg::OnDocOpenedArsctrl()
{
    VARIANT        var;
    BOOL           required;
    CScrollBar * pHorz, * pVert;
    CArsOle        * pArsCtrl;
    short          rc;

    pArsCtrl = (CArsOle*)GetDlgItem( IDC_ARSCtrl );

    rc = pArsCtrl->GetDocNumPages( &var );
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        DisplayMsg( rc, "GetDocNumPages" );
        return;
    }
    m_NumPages = var.lVal;

    rc = pArsCtrl->IsDocHorzScrollRequired( &var );
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        DisplayMsg( rc, "IsDocHorzScrollRequired" );
        return;
    }
    required = var.iVal;

    m_CurrentPage = 1;

    pHorz = (CScrollBar*)GetDlgItem( IDC_HORZ_SCROLLBAR );
    pVert = (CScrollBar*)GetDlgItem( IDC_VERT_SCROLLBAR );

    pHorz->ShowScrollBar( required );
    pVert->ShowScrollBar( TRUE );
    pHorz->SetScrollPos( 0 );
    pVert->SetScrollPos( 0 );
}

```

```

void CMainDlg::OnDocClosedArsctrl()
{
    CScrollBar * pBar;

    pBar = (CScrollBar*)GetDlgItem( IDC_HORZ_SCROLLBAR );
    if ( pBar != NULL )
        pBar->ShowScrollBar( FALSE );

    pBar = (CScrollBar*)GetDlgItem( IDC_VERT_SCROLLBAR );
    if ( pBar != NULL )
        pBar->ShowScrollBar( FALSE );
}

void CMainDlg::OnCloseDlg()
{
    short      rc;
    CArsOle * pArsCtrl;

    pArsCtrl = (CArsOle*)GetDlgItem( IDC_ARSCtrl );

    if( m_DocOpened )
        pArsCtrl->CloseDoc();

    rc = pArsCtrl->CloseFolder( );
    if ( rc != ARS_OLE_RC_SUCCESS )
        DisplayMsg( rc, "CloseFolder" );

    rc = pArsCtrl->Logoff( );
    if ( rc != ARS_OLE_RC_SUCCESS )
        DisplayMsg( rc, "Logoff" );

    EndDialog(0);
}

```

```

void CMainDlg::OnSetDocAttrs()
{
    CAttrsDlg dlg;
    dlg.DoModal( );
}

////////////////////////////////////
// CAttrsDlg dialog

CAttrsDlg::CAttrsDlg(CWnd* pParent /*=NULL*/)
: CDialog(CAttrsDlg::IDD, pParent)
{
   //{{AFX_DATA_INIT(CAttrsDlg)
        // NOTE: the ClassWizard will add member initialization here
   //}}AFX_DATA_INIT
}

void CAttrsDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(CAttrsDlg)
        // NOTE: the ClassWizard will add DDX and DDV calls here
   //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAttrsDlg, CDialog)
   //{{AFX_MSG_MAP(CAttrsDlg)
        ON_BN_CLICKED(IDC_BACK_COLOR, OnBackColor)
        ON_BN_CLICKED(IDC_IMAGE_COLOR, OnImageColor)
        ON_BN_CLICKED(IDC_ROTATION, OnRotation)
        ON_BN_CLICKED(IDC_ZOOM, OnZoom)
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()

```

```

////////////////////////////////////
// CAttrDlg message handlers

BOOL CAttrDlg::OnInitDialog()
{
    CArsOle * pArsCtrl;
    CListBox * pBackList, * pImageList;
    CEdit * pZoom, * pRotation;
    VARIANT var1, var2, var3;
    BOOL chg;
    short rc, j, back_color, image_color, zoom, rotation, min, max;
    int index;
    char data[10];

    pArsCtrl = (CArsOle*)pMainDlg->GetDlgItem( IDC_ARSCtrl );

    pBackList = (CListBox*)GetDlgItem( IDC_BACK_COLORS );
    pImageList = (CListBox*)GetDlgItem( IDC_IMAGE_COLORS );
    pZoom = (CEdit*)GetDlgItem( IDC_ZOOMPCT );
    pRotation = (CEdit*)GetDlgItem( IDC_ROT );

    rc = pArsCtrl->GetDocBackgroundColor( &var1, &var2 );
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        pMainDlg->DisplayMsg( rc, "GetDocBackgroundColor" );
        EndDialog( IDABORT );
        return TRUE;
    }
    back_color = var1.iVal;
    chg = var2.iVal;

    rc = pArsCtrl->GetDocImageColor( &var1, &var2 );
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        pMainDlg->DisplayMsg( rc, "GetDocImageColor" );
        EndDialog( IDABORT );
        return TRUE;
    }
    image_color = var1.iVal;
    chg = var2.iVal;

```



```

for ( j = 0; j < NUM_COLORS; j++ )
{
    index = pBackList->AddString( Colors[j].pText );
    pBackList->SetItemData( index, Colors[j].color );
    if ( Colors[j].color == back_color )
        pBackList->SetCurSel( index );
    index = pImageList->AddString( Colors[j].pText );
    pImageList->SetItemData( index, Colors[j].color );
    if ( Colors[j].color == image_color )
        pImageList->SetCurSel( index );
}

rc = pArsCtrl->GetDocZoom( &var1, &var2, &var3 );
if ( rc != ARS_OLE_RC_SUCCESS )
{
    pMainDlg->DisplayMsg( rc, "GetDocZoom" );
    EndDialog( IDABORT );
    return TRUE;
}
zoom = var1.iVal;
min = var2.iVal;
max = var3.iVal;

sprintf( data, "%d", (int)zoom );
pZoom->SetWindowText( data );

rc = pArsCtrl->GetDocRotation( &var1, &var2 );
if ( rc != ARS_OLE_RC_SUCCESS )
{
    pMainDlg->DisplayMsg( rc, "GetDocRotation" );
    EndDialog( IDABORT );
    return TRUE;
}
rotation = var1.iVal;
chg = var2.iVal;

sprintf( data, "%d", (int)rotation );
pRotation->SetWindowText( data );

return TRUE;
}

```

```

void CAttrsDlg::OnBackColor()
{
    CArsOle * pArsCtrl;
    CListBox * pList;
    VARIANT    var1, var2;
    BOOL        chg;
    short        rc, j, color;

    pArsCtrl = (CArsOle*)pMainDlg->GetDlgItem( IDC_ARSCtrl );
    pList = (CListBox*)GetDlgItem( IDC_BACK_COLORS );

    color = (short)pList->GetItemData( pList->GetCurSel( ) );

    rc = pArsCtrl->SetDocBackgroundColor( color );
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        pMainDlg->DisplayMsg( rc, "SetDocBackgroundColor" );
        rc = pArsCtrl->GetDocBackgroundColor( &var1, &var2 );
        if ( rc != ARS_OLE_RC_SUCCESS )
            pMainDlg->DisplayMsg( rc, "GetDocBackgroundColor" );
        else
        {
            color = var1.iVal;
            chg = var2.iVal;
            for ( j = 0; j < NUM_COLORS; j++ )
            {
                if ( (short)pList->GetItemData(j) == color )
                {
                    pList->SetCurSel(j);
                    break;
                }
            }
        }
    }
}

```

```

void CAttrsDlg::OnImageColor()
{
    CArsOle * pArsCtrl;
    CListBox * pList;
    VARIANT    var1, var2;
    BOOL        chg;
    short       rc, j, color;

    pArsCtrl = (CArsOle*)pMainDlg->GetDlgItem( IDC_ARSCtrl );
    pList = (CListBox*)GetDlgItem( IDC_IMAGE_COLORS );

    color = (short)pList->GetItemData( pList->GetCurSel( ) );

    rc = pArsCtrl->SetDocImageColor( color );
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        pMainDlg->DisplayMsg( rc, "SetDocImageColor" );
        rc = pArsCtrl->GetDocImageColor( &var1, &var2 );
        if ( rc != ARS_OLE_RC_SUCCESS )
            pMainDlg->DisplayMsg( rc, "GetDocImageColor" );
        else
        {
            color = var1.iVal;
            chg = var2.iVal;
            for ( j = 0; j < NUM_COLORS; j++ )
            {
                if ( (short)pList->GetItemData(j) == color )
                {
                    pList->SetCurSel(j);
                    break;
                }
            }
        }
    }
}

```

```

void CAttrsDlg::OnRotation()
{
    CArsOle * pArsCtrl;
    CEdit * pRotation;
    VARIANT var1, var2;
    BOOL chg;
    short rc, value;
    char data[10];

    pArsCtrl = (CArsOle*)pMainDlg->GetDlgItem( IDC_ARSCtrl );
    pRotation = (CEdit*)GetDlgItem( IDC_ROT );

    pRotation->GetWindowText( data, sizeof(data) );

    rc = pArsCtrl->SetDocRotation( (short)atoi( data ) );
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        pMainDlg->DisplayMsg( rc, "SetDocRotation" );
        rc = pArsCtrl->GetDocRotation( &var1, &var2 );
        if ( rc != ARS_OLE_RC_SUCCESS )
            pMainDlg->DisplayMsg( rc, "GetDocRotation" );
        else
        {
            value = var1.iVal;
            chg = var2.iVal;
            sprintf( data, "%d", (int)value );
            pRotation->SetWindowText( data );
        }
    }
    else
        OnZoom();
}

```

```

void CAttrsDlg::OnZoom()
{
    CArsOle * pArsCtrl;
    CEdit * pZoom;
    CScrollBar * pHorz, * pVert;
    VARIANT var1, var2, var3;
    BOOL required;
    short rc, value, min, max, horz_pos, vert_pos;
    char data[10];

    pArsCtrl = (CArsOle*)pMainDlg->GetDlgItem( IDC_ARSCtrl );
    pZoom = (CEdit*)GetDlgItem( IDC_ZOOMPCT );

    pZoom->GetWindowText( data, sizeof(data) );

    rc = pArsCtrl->SetDocZoom( (short)atoi( data ), &var1, &var2 );
    horz_pos = var1.iVal;
    vert_pos = var2.iVal;
    if ( rc != ARS_OLE_RC_SUCCESS )
    {
        pMainDlg->DisplayMsg( rc, "SetDocZoom" );
        rc = pArsCtrl->GetDocZoom( &var1, &var2, &var3 );
        if ( rc != ARS_OLE_RC_SUCCESS )
            pMainDlg->DisplayMsg( rc, "GetDocZoom" );
        else
        {
            value = var1.iVal;
            min = var2.iVal;
            max = var3.iVal;
            sprintf( data, "%d", (int)value );
            pZoom->SetWindowText( data );
        }
    }
    else
    {
        rc = pArsCtrl->IsDocHorzScrollRequired( &var1 );
        if ( rc != ARS_OLE_RC_SUCCESS )
        {
            pMainDlg->DisplayMsg( rc, "IsDocHorzScrollRequired" );
            return;
        }
        required = var1.iVal;

        pHorz = (CScrollBar*)pMainDlg->GetDlgItem( IDC_HORZ_SCROLLBAR );
        pVert = (CScrollBar*)pMainDlg->GetDlgItem( IDC_VERT_SCROLLBAR );

        pHorz->ShowScrollBar( required );
        pHorz->SetScrollPos( horz_pos );
        pVert->SetScrollPos( vert_pos );
    }
}

```

Appendix E. Accessibility features

The OnDemand product includes a number of features that make the product more accessible for people with disabilities. These features include:

- Features that facilitate keyboard input and navigation
- Features that enhance display properties
- Options for audio and visual alert cues
- Compatibility with assistive technologies
- Compatibility with accessibility features of the operating system
- Accessible documentation formats

Keyboard input and navigation

Keyboard input

The OnDemand clients can be operated using only the keyboard. Menu items and controls provide access keys that allow users to activate a control or select a menu item directly from the keyboard. These keys are self-documenting, in that the access keys are underlined on the control or menu where they appear.

Keyboard focus

In Windows-based systems, the position of the keyboard focus is highlighted, indicating which area of the window is active and where the user's keystrokes will have an effect.

Features for accessible display

The OnDemand clients have a number of features that enhance the user interface and improve accessibility for users with low vision. These accessibility enhancements include support for high-contrast settings and customizable font properties.

High-contrast mode

The OnDemand clients support the high-contrast-mode option that is provided by the operating system. This feature assists users who require a higher degree of contrast between background and foreground colors.

Font settings

In Windows-based systems, the user can specify display settings that determine the color, size, and font for the text in menus and dialog windows. The OnDemand client allows the user to select the font for the document list.

Non-dependence on color

Users do not need to distinguish between colors in order to use any of the functions in this product.

Alternative alert cues

In Windows-based systems, the SoundSentry feature can be used to provide visual feedback for general application and system alerts such as warning beeps.

Compatibility with assistive technologies

The OnDemand clients are compatible with screen reader applications such as Narrator and Via Voice. The OnDemand clients have the properties required for these accessibility applications to make onscreen information available to blind users.

Accessible documentation

Documentation for the OnDemand product is available in HTML format. This allows users to view documentation according to the display preferences set in their browsers. It also allows the use of screen readers and other assistive technologies.

Appendix F. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only the IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe on any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department MYGA Building 001L
6300 Diagonal Highway
Boulder, CO 80301-9191
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to IBM's application programming interfaces.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks and Service Marks

Advanced Function Presentation™, AFP™, AIX®, DB2®, DB2 Universal Database®, IBM®, iSeries™, Infoprint™, MVS®, OS/390®, Print Services Facility™, and z/OS™ are trademarks of International Business Machines Corporation in the United States, other countries, or both.

Adobe, the Adobe logo, Acrobat and the Acrobat logo are trademarks of Adobe Systems Incorporated, which may be registered in certain jurisdictions.

Microsoft®, Windows®, and Windows NT® are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX® is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

Portions of the OnDemand Windows client program contain licensed software from Pixel Translations Incorporated, © Pixel Translations Incorporated 1990, 2002. All rights reserved.

Other company, product, and service names may be trademarks or service marks of others.

Index

Special Characters

- [CHARSET] section
 - attributes
 - fgid 311
 - font global identifier (fgid) 311
 - font height 311
 - font width 311
 - strikeover 311
 - underline 311
- [CODEPG] section 314
 - attributes
 - code page global identifier 315
 - CPGID 315
 - possible values 315
 - Windows character set 315
- [FGID] section
 - attributes
 - font type family 312
 - italic 312
 - style 312
 - weight 312

A

- about this publication xi
- accessibility 401
- ACTIVATE_DOC command 199
- ACTIVATE_FOLDER command 200
- alias file
 - [CHARID] section 319
- ALIAS.FNT 318
- ANNOTATE_DOC command 201
- ARRANGE_DOCS command 203
- attributes
 - code page global identifier
 - possible values 315
 - shipped default 315
 - Windows character set
 - ANSI 315
 - possible values 315
 - shipped default 315
 - SYMBOL 315

B

- BLDCPMAP.REX 317

C

- Change Password parameter 186
- CHANGE_PASSWORD command 204
- character set definition file
 - [CHARSET] section 311
 - [FGID] section 312
 - attributes
 - fgid 311
 - font global identifier (fgid) 311
 - font height 311
 - font type family 312
 - font width 311
 - italic 312
 - strikeover 311
 - style 312
 - underline 311
 - weight 312
- CID 303
- CLEAR_FIELDS command 205
- client
 - distributing user-defined files 299
 - installing on a network 293
 - integrating with Monarch 281
 - network installation 293
 - user-defined files 299
- CLOSE_ALL_DOCS command 206
- CLOSE_ALL_FOLDERS command 207
- CLOSE_DOC command 208
- CLOSE_FOLDER command 209
- code page definition file 315
 - [CODEPG] section 314
 - code page global identifier 314
- code page global identifier 314
 - possible values 315
 - shipped default 315
- code page map file
 - AFP code page 315
 - BLDCPMAP.REX 317
 - REXX program 317
- code page map file REXX program 317
- coded font file

coded font file (*continued*)

CODED.FNT 310

ICODED.FNT 310

coded font file rules 311

command line reference

parameter syntax 185

parameters 185

Change Password /C 186

Disable Anticipation /V 188

Disable Close Folder /Z 188

Disable Exit — /K 187

Disable Logoff and Password Change /X 188

Disable Update Servers /Y 188

Disable User Confirmation /B 188

Enable DDE Interface /I 187

Folder Name /F 186

Free Memory when Folder Closed /Q 188

Language Path /L 188

Logon Password /P 186

Logon Server Name /S 186

Logon User ID /U 186

Maximum Open Folders /O 187

Product Title /T 185

Window Placement /W 187

Starting OnDemand 32-bit client 185

compact install 293

configuration, installation, and distribution 303

COPY_DOC_PAGES command 210

copying software to a file server 294

custom install 293

customization overview 183

D

data mining

with Monarch 281

DDE commands

ACTIVATE_DOC 199

ACTIVATE_FOLDER 200

ANNOTATE_DOC 201

ARRANGE_DOCS 203

CHANGE_PASSWORD 204

CLEAR_FIELDS 205

CLOSE_ALL_DOCS 206

CLOSE_ALL_FOLDERS 207

CLOSE_DOC 208

CLOSE_FOLDER 209

COPY_DOC_PAGES 210

DELETE_DOC 212

DESELECT_DOC 213

DDE commands (*continued*)

DISABLE_SWITCH 214

ENABLE_SWITCH 215

EXIT 216

GET_DISPLAY_FIELDS 217

GET_DOC_VALUES 218

GET_FOLDER_FIELDS 220

GET_FOLDERS 221

GET_NUM_DOC_PAGES 223

GET_NUM_DOCS_IN_LIST 222

GET_PRINTERS 224

GET_QUERY_FIELDS 225

GET_SERVERS 226

LOGOFF 227

LOGON 228

OPEN_DOC 230

OPEN_FOLDER 232

PRINT_DOC 237

RESTORE_DEFAULTS 240

RETRIEVE_DOC 241

return codes 256

SEARCH_FOLDER 243

SELECT_DOC 244

SET_FIELD_DATA 245

SET_FOCUS 247

SET_HELP_PATH 248

SET_USER_MSG_MODE 249

SHOW_WINDOW 250

STORE_DOC 251

UPDATE_DOC 254

DDEML

termination 194

transactions 195

DELETE_DOC command 212

DESELECT_DOC command 213

directories 294

disability 401

Disable Anticipation parameter 188

Disable Close Folder parameter 188

Disable Exit parameter 187

Disable Logoff and Password Change parameter 188

Disable Update Servers parameter 188

Disable User Confirmation parameter 188

DISABLE_SWITCH command 214

distributing user-defined files 299

distribution install 293, 294

distribution server 281, 293

Dynamic Data Exchange interface reference 191

E

Enable DDE Interface parameter 187
ENABLE_SWITCH command 215
EXIT command 216

F

fgid 311
file rules
 alias file 319
 character set definition file 314
 code page definition file 315
 code page map file 316
 coded font 311
file server
 copying OnDemand software to 294
 installing OnDemand clients on 296
files
 alias file 318
 character set definition file
 fgid 311
 font global identifier (fgid) 311
 font height 311
 font width 311
 strikeover 311
 underline 311
 code page definition file 314
 code page map file 315
 coded font 310
files supplied for mapping fonts
 alias file 308
 ALIAS.FNT 308
 character set definition file 308
 code page definition file 308
 code page map file 308
 coded font files 308
 CODED.FNT 308
 CPDEF.FNT 308
 cpgid.CP 308
 CSDEF.FNT 308
 ICODED.FNT 308
 syntax rules 310
Folder Name parameter 186
font global identifier (fgid) 311
font height 311
font mapping files, syntax rules for 310
font type family 312
font width 311
fonts

fonts (*continued*)

 code page map file REXX program 317
 file rules
 alias file 319
 character set definition file 314
 code page definition file 315
 code page map file 316
 coded font 311
 files
 alias file 318
 character set definition file 311
 code page definition file 314
 code page map file 315
 coded font 310
 files for mapping 308
 syntax rules 310
 IBM Core Interchange
 Symbols 307
 mapping
 need for 307
 steps for 309
 TrueType support 320
Free Memory when Folder Closed parameter 188

G

GET_DISPLAY_FIELDS command 217
GET_DOC_VALUES command 218
GET_FOLDER_FIELDS command 220
GET_FOLDERS command 221
GET_NUM_DOC_PAGES command 223
GET_NUM_DOCS_IN_LIST command 222
GET_PRINTERS command 224
GET_QUERY_FIELDS command 225
GET_SERVERS command 226

I

IBM Core Symbols fonts
 roman bold 307
 roman medium 307
IBM trademarks used in this book 405
install
 compact 293
 custom 293
 distribution 293, 294
 local 293
 multiple user 293, 296, 297
 node 293
 standard 293

- install (*continued*)
 - typical 293
 - user-defined files 295, 297
- installation
 - creating and using response files 303
 - response files 303
- installation directories 294
- installing
 - clients on a network 293
 - Monarch 281
 - network file server 293
 - user-defined files 299
- installing clients on a network user's PC 297
- installing software on a file server 296
- integration with Monarch 281
- interface reference for DDE 191
- italic 312

K

- keyboard 401

L

- Language Path parameter 188
- legal notices 403
- local install 293
- LOGOFF command 227
- LOGON command 228
- Logon Password parameter 186
- Logon Server Name parameter 186
- Logon User ID parameter 186

M

- mapping fonts
 - about 307
 - file rules
 - alias file 319
 - character set definition file 314
 - code page definition file 315
 - code page map file 316
 - coded font 311
 - files
 - alias file 318
 - character set definition file 311
 - code page definition file 314
 - code page map file 315
 - coded font 310
 - files supplied for
 - alias file 308

- mapping fonts (*continued*)
 - files supplied for (*continued*)
 - ALIAS.FNT 308
 - character set definition file 308
 - code page definition file 308
 - code page map file 308
 - coded font files 308
 - CODED.FNT 308
 - CPDEF.FNT 308
 - cpgid.CP 308
 - CSDEF.FNT 308
 - ICODED.FNT 308
 - syntax rules 310
 - need for 307
 - program
 - code page map file REXX 317
 - steps for 309
- Maximum Open Folders parameter 187
- Monarch 281
 - integration with 281
- multiple user install 293, 296

N

- need for mapping fonts 307
- network file server 293
 - copying OnDemand software to 294
 - installing OnDemand clients on 296
- network installation 293
- network user
 - installing OnDemand clients 297
- node install 293, 297
- NOMATCH 314
- notices 403

O

- OnDemand customization overview 183
- OnDemand DDE commands
 - ACTIVATE_DOC 199
 - ACTIVATE_FOLDER 200
 - ANNOTATE_DOC 201
 - ARRANGE_DOCS 203
 - CHANGE_PASSWORD 204
 - CLEAR_FIELDS 205
 - CLOSE_ALL_DOCS 206
 - CLOSE_ALL_FOLDERS 207
 - CLOSE_DOC 208
 - CLOSE_FOLDER 209
 - COPY_DOC_PAGES 210

OnDemand DDE commands (*continued*)

- DELETE_DOC 212
- DESELECT_DOC 213
- DISABLE_SWITCH 214
- ENABLE_SWITCH 215
- EXIT 216
- GET_DISPLAY_FIELDS 217
- GET_DOC_VALUES 218
- GET_FOLDER_FIELDS 220
- GET_FOLDERS 221
- GET_NUM_DOC_PAGES 223
- GET_NUM_DOCS_IN_LIST 222
- GET_PRINTERS 224
- GET_QUERY_FIELDS 225
- GET_SERVERS 226
- LOGOFF 227
- LOGON 228
- OPEN_DOC 230
- OPEN_FOLDER 232
- PRINT_DOC 237
- RESTORE_DEFAULTS 240
- RETRIEVE_DOC 241
 - return codes 256
- SEARCH_FOLDER 243
- SELECT_DOC 244
- SET_FIELD_DATA 245
- SET_FOCUS 247
- SET_HELP_PATH 248
- SET_USER_MSG_MODE 249
- SHOW_WINDOW 250
- STORE_DOC 251
- UPDATE_DOC 254
- OPEN_DOC command 230
- OPEN_FOLDER command 232
- other companies trademarks used in this book 405

P

- parameter syntax 185
- PRINT_DOC command 237
- Product Title parameter 185

R

- reference workstation 281
- report mining
 - with Monarch 281
- response files 303
- RESTORE_DEFAULTS command 240

- RETRIEVE_DOC command 241
- return codes 256
- REXX program, code page map file 317
- rules
 - character set definition file 314
 - code page definition file 315
 - coded font file 311
 - for font definition file syntax 310

S

- SEARCH_FOLDER command 243
- SELECT_DOC command 244
- server
 - copying OnDemand software to 294
 - installing OnDemand clients on 296
- service marks used in this book
 - IBM 405
 - other companies 405
- SET_FIELD_DATA command 245
- SET_FOCUS command 247
- SET_HELP_PATH command 248
- SET_USER_MSG_MODE command 249
- SHOW_WINDOW command 250
- standard install 293
- Starting OnDemand 32-bit 185
- steps for mapping fonts 309
- STORE_DOC command 251
- strikeover 311
- style 312
- syntax rules for command line parameters 185
- syntax rules for font mapping files 310

T

- terminating the DDE conversation 194
- trademarks used in this book
 - IBM 405
 - other companies 405
- transactions
 - DDEML 195
 - example 195
- TrueType font support 320
- typical install 293

U

- underline 311
- UPDATE_DOC command 254

user-defined files 295, 297, 299

W

weight 312

Window Placement parameter 187

Windows character set

- ANSI 315

- possible values 315

- shipped default 315

- SYMBOL 315

Windows client

- CID 303

- configuration, installation, and distribution 303

- creating and using response files 303

- distributing user-defined files 299

- installing on a network 293

- integrating with Monarch 281

- network installation 293

- response files 303

- user-defined files 299

Readers' Comments — We'd Like to Hear from You

**IBM Content Manager OnDemand
Windows Client Customization Guide
Publication No. SC27-0837-01**

Use this form to provide comments about this publication, its organization, or subject matter. Understand that IBM may use the information any way it believes appropriate, without incurring any obligation to you. Your comments will be sent to the author's department for the appropriate action. Comments may be written in your language.

Note: IBM publications are not stocked at the location to which this form is addressed. Direct requests for publications or for assistance in using your IBM system, to your IBM representative or local IBM branch office.

	Yes	No
• Does the publication meet your needs?	_____	_____
• Did you find the information:		
Accurate?	_____	_____
Easy to read and understand?	_____	_____
Easy to retrieve?	_____	_____
Organized for convenient use?	_____	_____
Legible?	_____	_____
Complete?	_____	_____
Well illustrated?	_____	_____
Written for your technical level?	_____	_____
• Do you use this publication:		
As an introduction to the subject?	_____	_____
As a reference manual?	_____	_____
As an instructor in class?	_____	_____
As a student in class?	_____	_____
• What is your occupation?	_____	_____

Thank you for your input and cooperation.

Note: You may either send your comments by fax to 1-303-924-7522 or mail your comments. If mailed in the U.S.A., no postage stamp is necessary. For residents outside the U.S.A., your local IBM office or representative will forward your comments.

Comments:

Name

Address

Company or Organization

Phone No.

Readers' Comments — We'd Like to Hear from You
SC27-0837-01



Cut or Fold
Along Line

Fold and Tape

Please do not staple

Fold and Tape



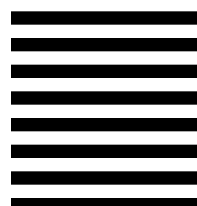
NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

Information Development
IBM Corporation
Department MYGA Building 001L
P O Box 1900
BOULDER CO USA 80301-9817



Fold and Tape

Please do not staple

Fold and Tape

SC27-0837-01

Cut or Fold
Along Line



Program Number: 5655-H39
5697-G34
5722-RD1



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC27-0837-01



Spine information:



IBM Content Manager OnDemand

Windows Client Customization Guide